File Handling in C++

Files (Streams)

- Files are used to store data in a relatively permanent form, on floppy disk, hard disk, tape or other form of secondary storage. Files can hold huge amounts of data if need be.

- Ordinary variables (even records and arrays) are kept in main memory which is temporary and rather limited in size. The following is a comparison of the two types of storage:

Data File Handling In C++

- **File:** The information / data stored under a specific name on a storage device, is called a file.
- **Stream:** It refers to a sequence of bytes.
- **Text file:** It is a file that stores information in ASCII characters. In text files, each line of text is terminated with a special character known as EOL (End of Line) character or delimiter character. When this EOL character is read or written, certain internal translations take place.
- **Binary file:** It is a file that contains information in the same format as it is held in memory. In binary files, no delimiters are used for a line and no translations occur here.

C++ Streams

- A Stream is a general name given to flow of data.
- Different streams are used to represent different kinds of data flow.
- Each stream is associated with a particular class, which contains member functions and definitions for dealing with that particular kind of data flow.

File Handling in C++

- **File Handling** concept in C++ language is used for store a data permanently in computer. Using file handling we can store our data in Secondary memory (Hard disk).
- **Why use File Handling**
  - Memory is **volatile**
  - Any data that you key in by keyboard while a program is running is also **volatile**
    - For **permanent storage**.
    - The transfer of input - data or output - data from one computer to another can be easily done by using files.

Header File for File Handling

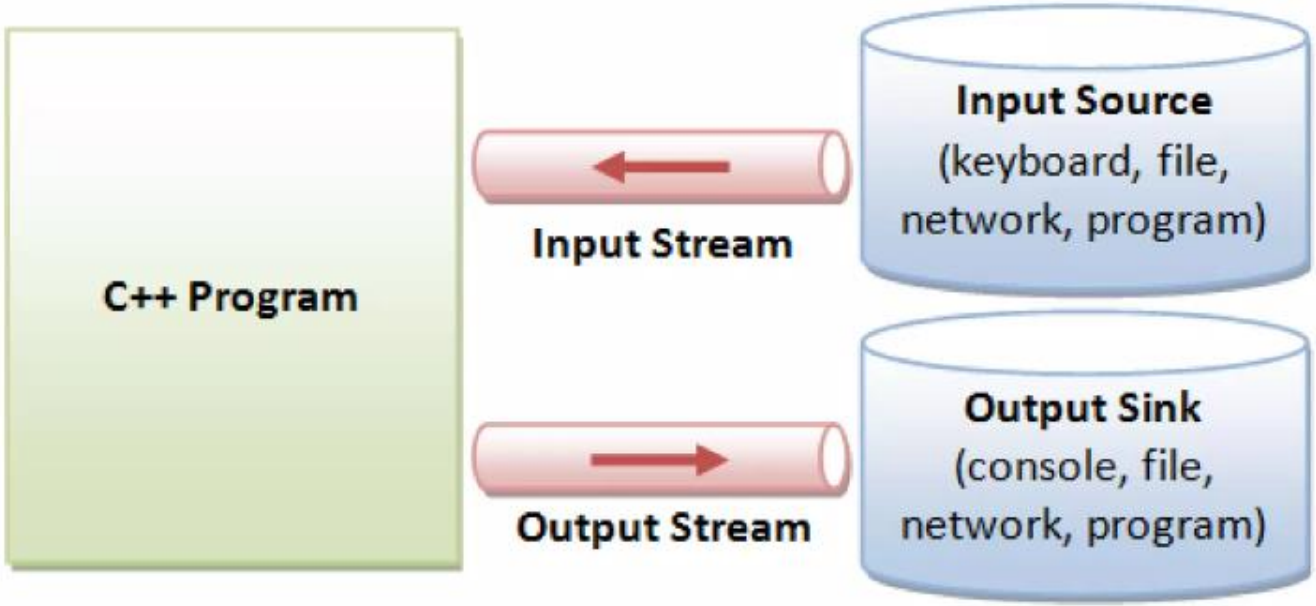
- So far, we have been using the **iostream** standard library, which provides **cin** and **cout** methods for reading from standard input and writing to standard output respectively.
- This tutorial will teach you how to read and write from a file. This requires another standard C++ library called **fstream**, which defines three new data types:

Datatype	Description
ofstream	This is used to create a file and write data on files
ifstream	This is used to read data from files
fstream	This is used to both read and write data from/to files

Header File for File Handling

- Input File Stream #include<ifstream>
- Output file stream #include<ofstream>
- Both (Input & Output) #include<fstream>

File Handling in C++



Internal Data Formats:

- Text: `char`, `wchar_t`
- `int`, `float`, `double`, etc.

External Data Formats:

- Text in various encodings (US-ASCII, ISO-8859-1, UCS-2, UTF-8, UTF-16, UTF-16BE, UTF16-LE, etc.)
- Binary (raw bytes)

How to Achieve File Handling

- **Create** a **Handler** Object
- **Naming** a **file**
- **Opening** a **file**
- **Reading** data from **file**
- **Writing** data into **file**
- **Closing** a **file**

Reading a File

```
#include<fstream>

ifstream inputFile ;

inputFile.open("Record.txt");

c:\myProg\Record.txt

inputFile.close();
```

Reading a File

```
#include<fstream>

ifstream inputFile ;

inputFile.open("Record.txt");

c:\myProg\Record.txt

inputFile.close();
```

Handler

Opening the File

Closing the File

Functions use in File Handling

Function	Operation
open()	To create a file
close()	To close an existing file
get()	Read a single character from a file
put()	write a single character in file.
read()	Read data from file
write()	Write data into file.

Defining and Opening a File

- The function open() can be used to open multiple files that use the same stream object.

- **Syntax**

```
file-stream-class   stream-object;
stream-object.open ("filename");
```

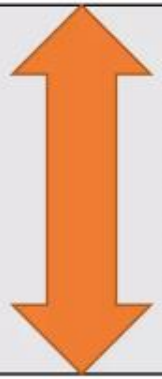
- **Example**

```
ofstream  inputFile;      // create stream
inputFile . open ("data1.txt");  // connect stream to data1
```

Defining and Opening a File

```
ofstream  inputFile;      // create stream
inputFile . open ("data1.txt"); // connect stream to data1
```

- **Opening File Using Constructor**



Both are Same: Creating and writing data in file

```
ofstream inputFile ("data1.txt");
```

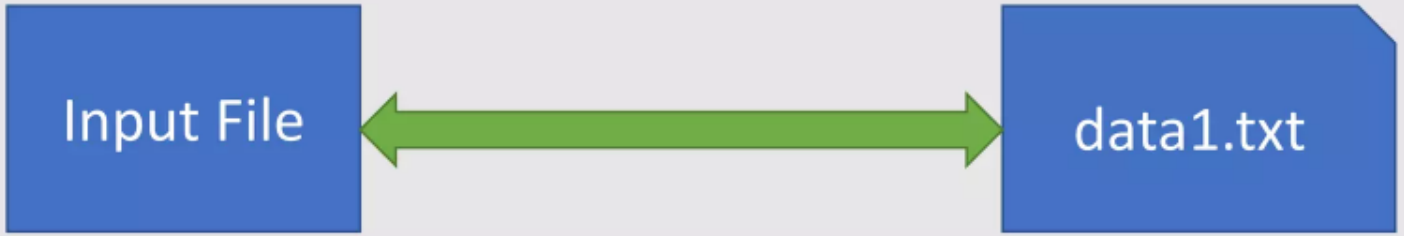
Closing a File

- A file must be close after completion of all operation related to file. For closing file we need **close()** function.

```
inputFile.close();
```

Opening and Closing File

Process of Opening `inputFile.open("data1.txt");`



Process of Closing `inputFile.close("data1.txt");`

File Opening Mode

File Mode Parameter	Meaning
ios::app	Append mode. All output to that file to be appended to the end.
ios::ate	Open a file for output and move the read/write control to the end of the file.
ios::binary	file open in binary mode
ios::in	open file for reading only
ios::out	open file for writing only
ios::nocreate	open fails if the file does not exist
ios::noreplace	open fails if the file already exist
ios::trunc	delete the contents of the file if it exist

File Opening Mode

- The default value for **fstream** mode parameter is **in | out**. It means that file is opened for reading and writing when you use **fstream** class.

- When you use **ofstream** class, default value for mode is **out** and the default value for **ifstream** class is **in**.

File Opening Mode

- Both ios :: app and ios :: ate take us to the end of the file when it is opened. The difference between the two parameters is that the ios :: app allows us to add data to the end of file only, while ios :: ate mode permits us to add data or to modify the existing data any where in the file.
- The mode can combine two or more parameters using the bitwise OR operator (symbol |)

```
fstream file;
file.Open("data1 . txt", ios :: out | ios :: in);
```

File Handling in C++

We can read data from file and write data to file in three ways.

- Reading or writing characters using get() and put() member functions.
- Reading or writing formatted I/O using insertion operator (<<) and extraction operator (>>).
- Reading or writing object using read() and write() member functions.

Input And Output Operation

- **put() and get() function**
  the function put() writes a single character to the associated stream. Similarly, the function get() reads a single character form the associated stream. **Example :**

```
file.get(ch);
file.put(ch);
```

- **write() and read() function**
  write() and read() functions write and read blocks of binary data. **Example:**

```
file.read((char *)&obj, sizeof(obj));
file.write((char *)&obj, sizeof(obj));
```

Error Checking

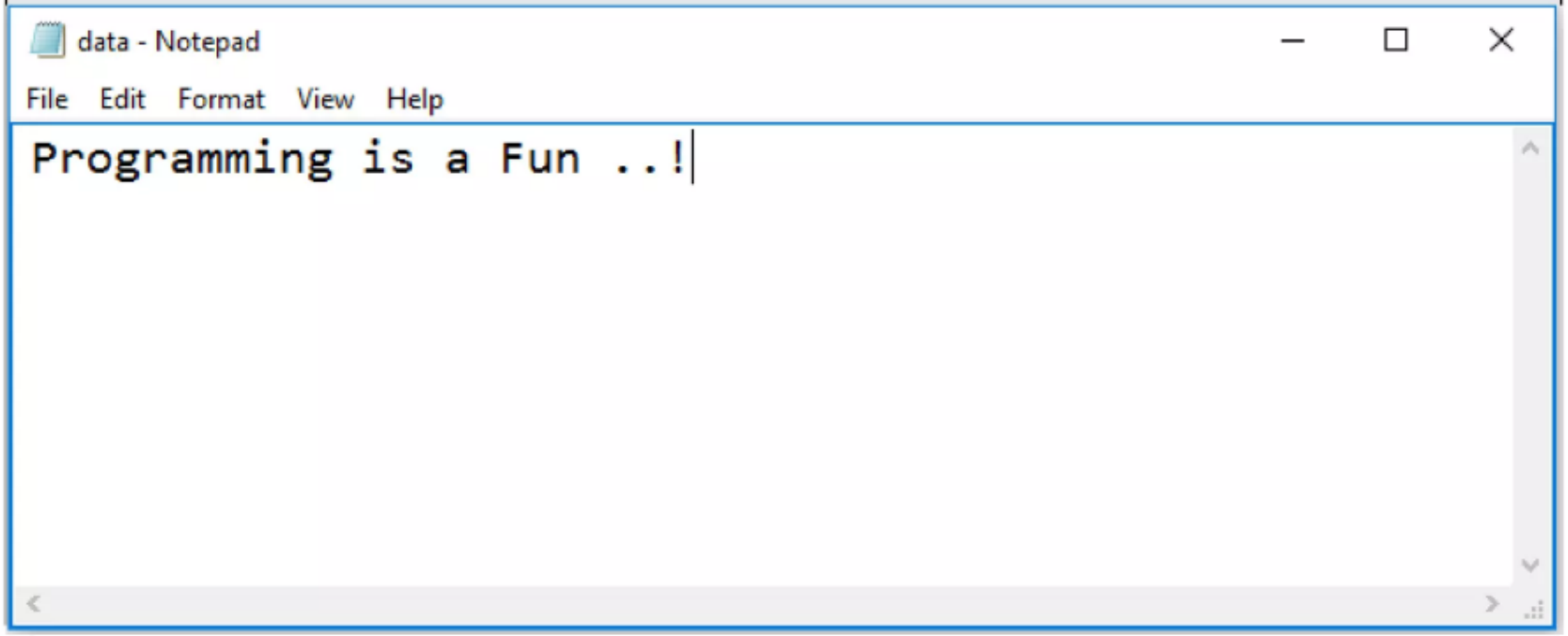
```
ifstream myFile;
myFile.open("File.txt", ios::in);

if (!myFile)
{
        cout << "The file cannot open" ;
}
```

Error Handling Functions

Function	Return Value And Meaning
eof()	returns true (non zero) if end of file is encountered while reading; otherwise return false(zero)
fail()	return true when an input or output operation has failed
bad()	returns true if an invalid operation is attempted or any unrecoverable error has occurred.
good()	returns true if no error has occurred.

Example No. 2

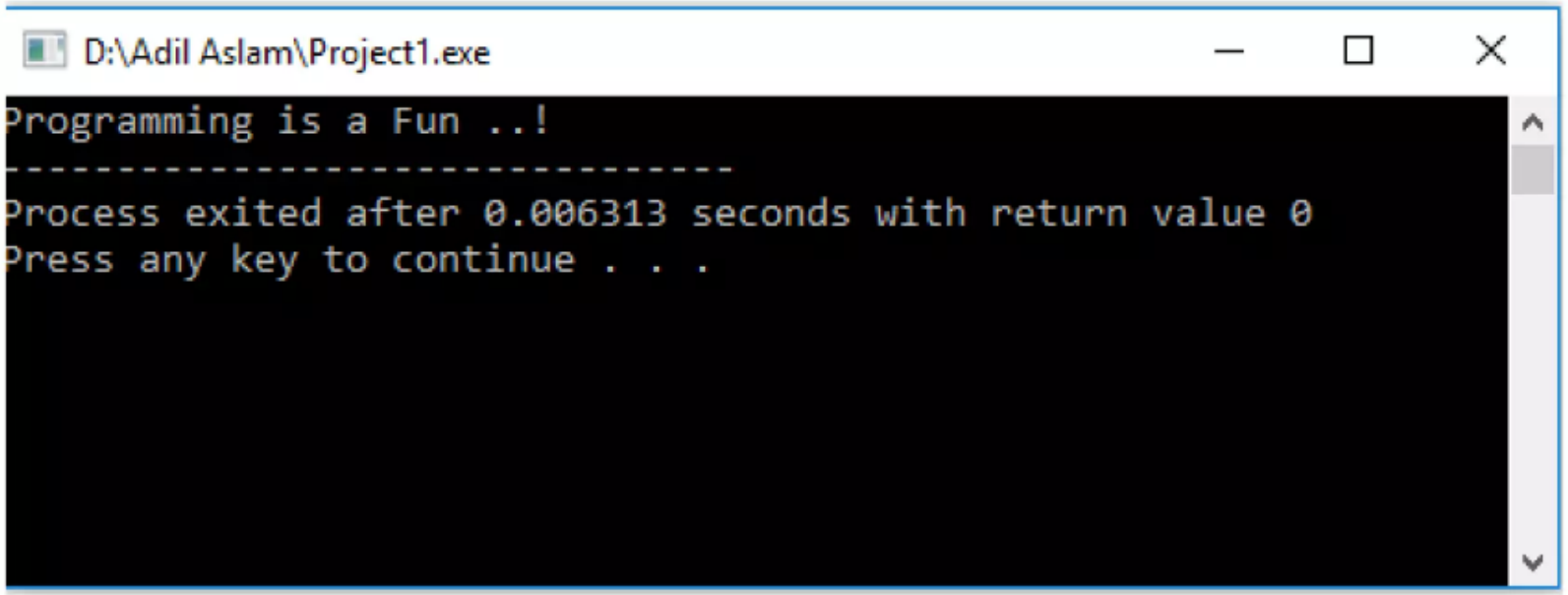
Read the Following File .
File Name is “data.txt”.



Read From Text File and Display It

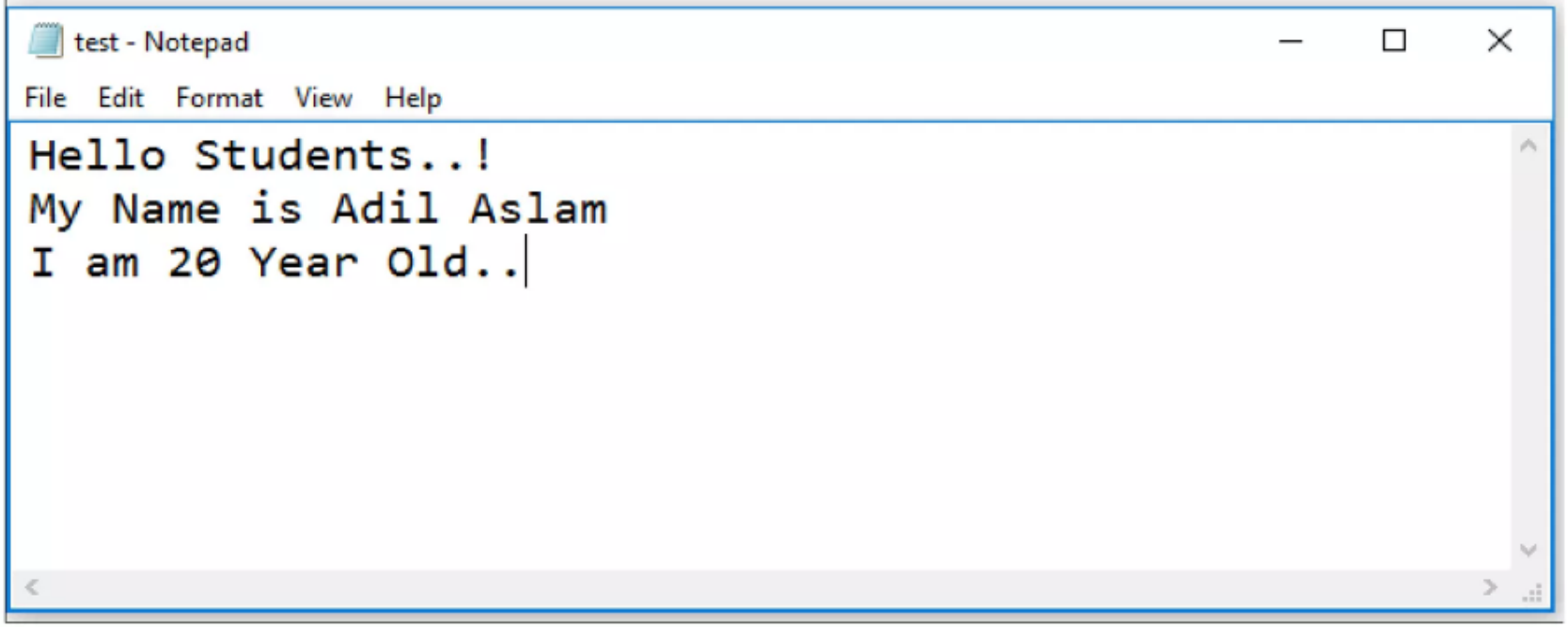
```
#include<iostream>
#include <fstream>
using namespace std;
int main() {
        ifstream input;  string str;
        input.open ("data.txt");
        if (! input) {
                cout << "Sorry, file can not be open!!!" << endl;
        }
        else {
                while (! input.eof()) {
                        input >> str;
                        cout << str << " ";
                }
        }
}
```

Output of the Previous Program is :



Example No. 2

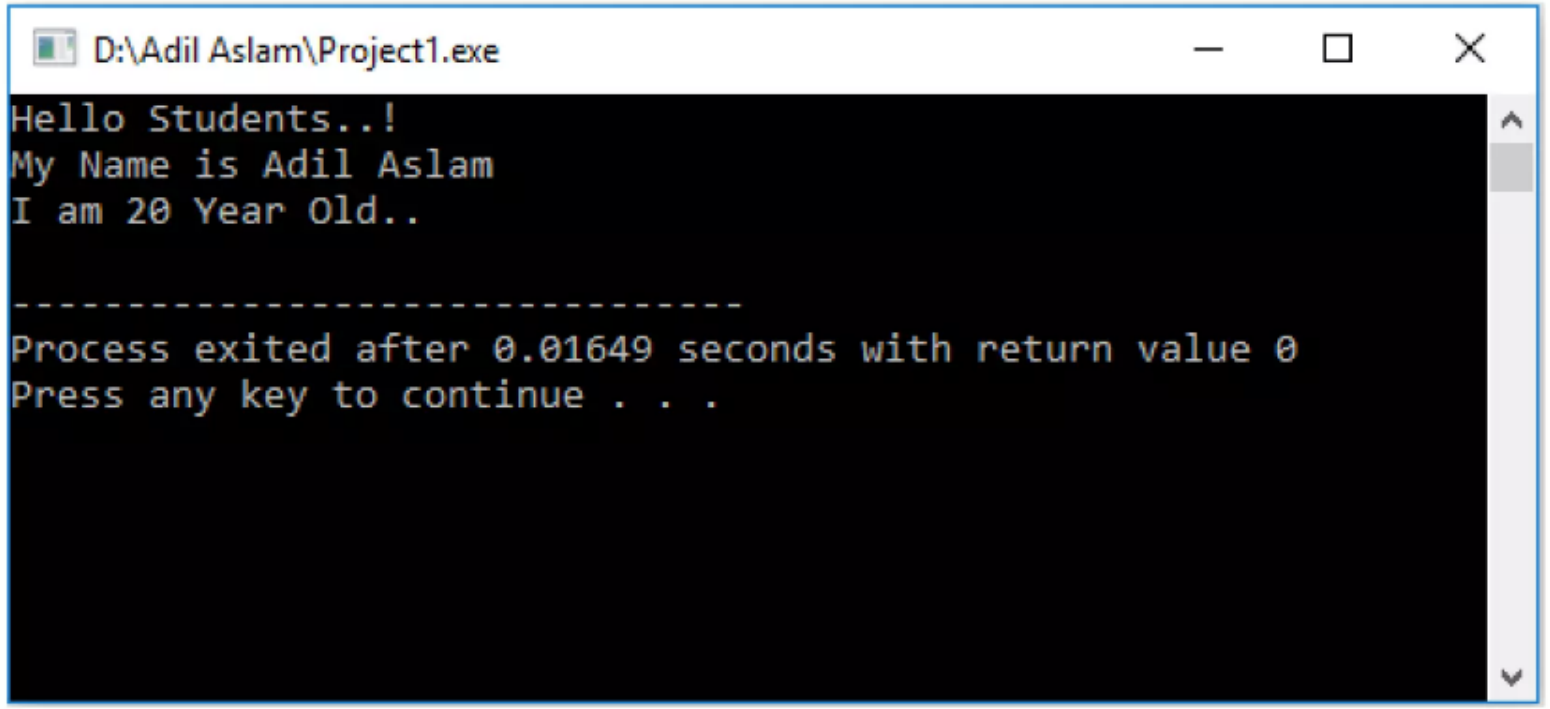
Read the Following File .
File Name is “test.txt”.



Read From Text File and Display It

```
#include <iostream>
#include<fstream>
using namespace std;
int main()  {
            ifstream input("test.txt");
            string line;
            if(!input) {
        cout << "Cannot open input file.\n";
   return 1;
 }
            /* While there is still a line. */
            while(getline(input, line)) {
            /* Printing goes here. */
                        cout << line << endl;
            }
            input.close();
}
```

Output of the Previous Program is :



Example No. 3

Read the Following File .
File Name is “Students.txt”.

Students - Notepad

File Edit Format View Help

```
Name     Age      Discipline
----     ---      ----------
Adil     20       BSCS
Hina     20       MSIT
Waqar    21       MSCS
Ali      22       MSSE
```

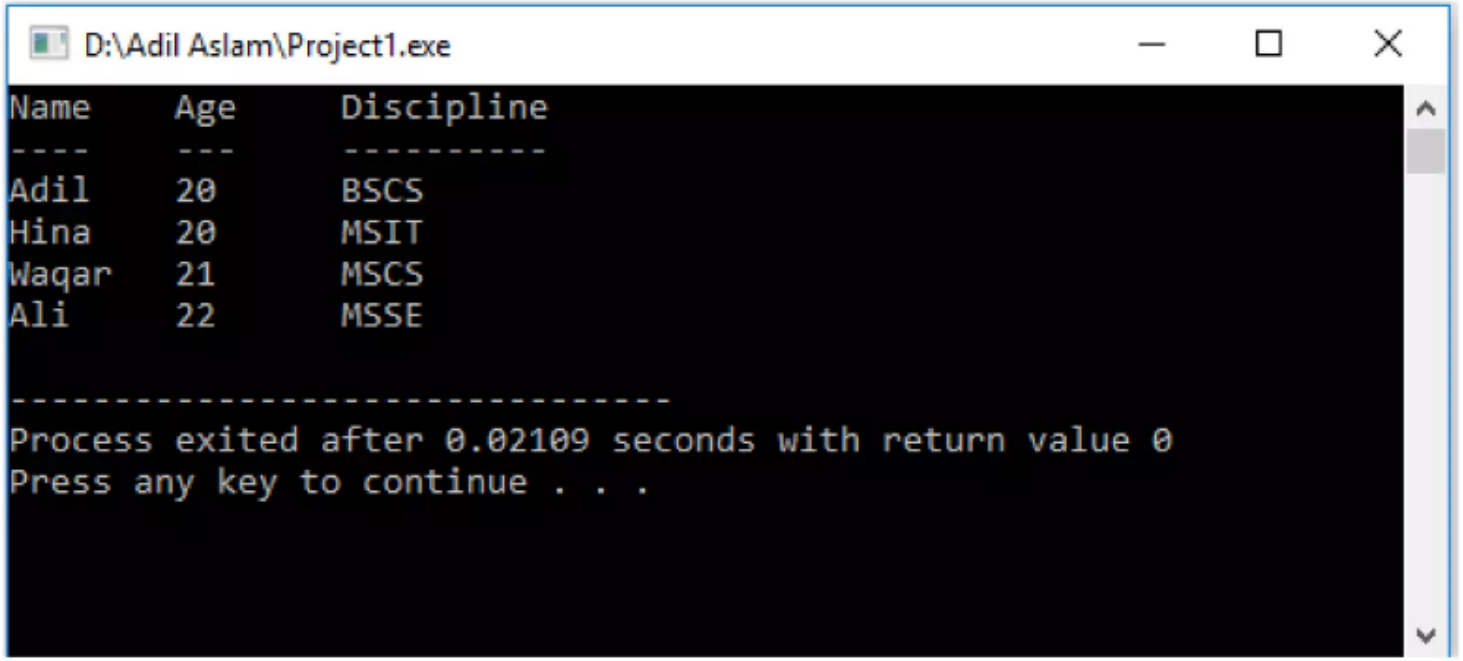
Read From Text File and Display It-1

```
#include<iostream>
#include <fstream>
using namespace std;
int main()
{
        string name, age, disc;
        ifstream inputFile;
        inputFile.open("Students.txt");
        if (! inputFile)
        {
                cout << "Sorry, file can not be opened!" ;
        }
```

Read From Text File and Display It-2

```
        else
        {
                while (! inputFile.eof())
                {
                        inputFile >> name >> age >> disc;
                        cout << name << "\t" << age << "\t"
                        <<disc;
                        cout << endl;
                }
        }
} //end of main
```

Output of the Previous Program is :



Read From Text File and Display It-1

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
        ifstream input("Students.txt");
        string line;
        if(!input) {
       cout << "Cannot open input file.\n";
   return 1;
  }
```

Read From Text File and Display It-2

```
        /* While there is still a line. */
        while(getline(input, line)) {
        /* Printing goes here. */
                cout << line << endl;
        }

        input.close();
}
```

Students - Notepad

File Edit Format View Help

```
Name     Age      Discipline
----     ---      ----------
Adil     20       BSCS
Hina     20       MSIT
Waqar    21       MSCS
Ali      22       MSSE
```

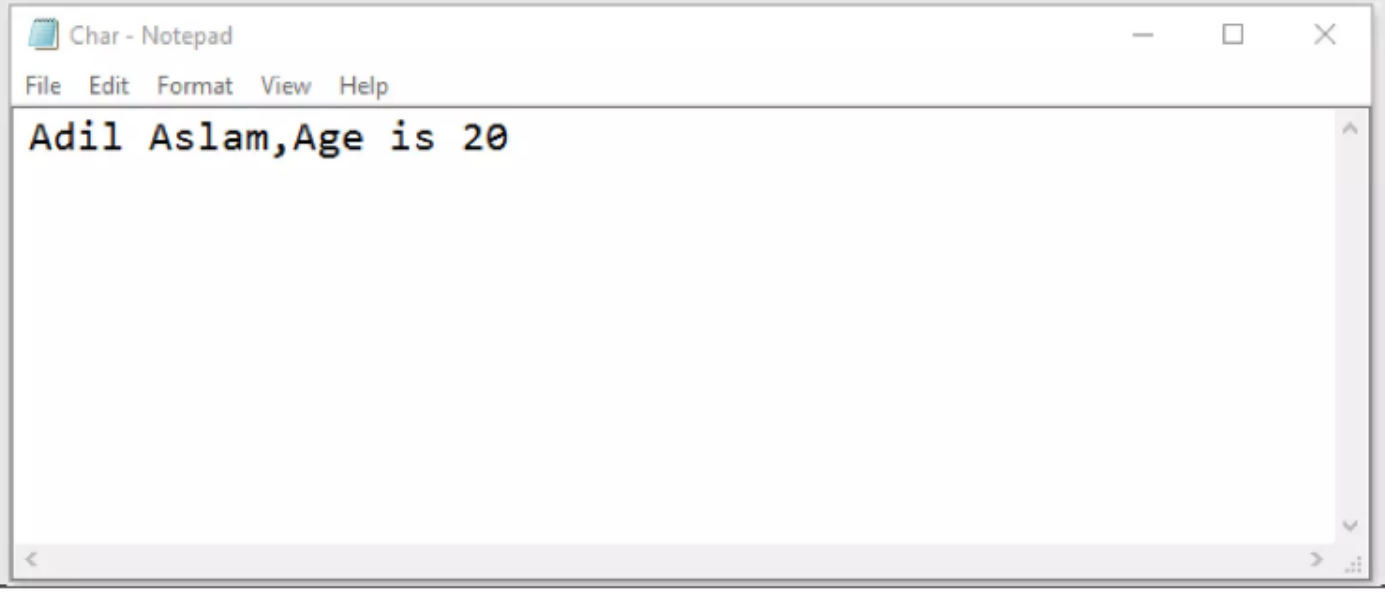
D:\Adil Aslam\Project1.exe

```
Name     Age      Discipline
----     ---      ----------
Adil     20       BSCS
Hina     20       MSIT
Waqar    21       MSCS
Ali      22       MSSE

--------------------------------
Process exited after 0.02109 seconds with return value 0
Press any key to continue . . .
```

Example No. 4

Read the Following File .
File Name is “Char.txt”.

Count Number of Characters in this File



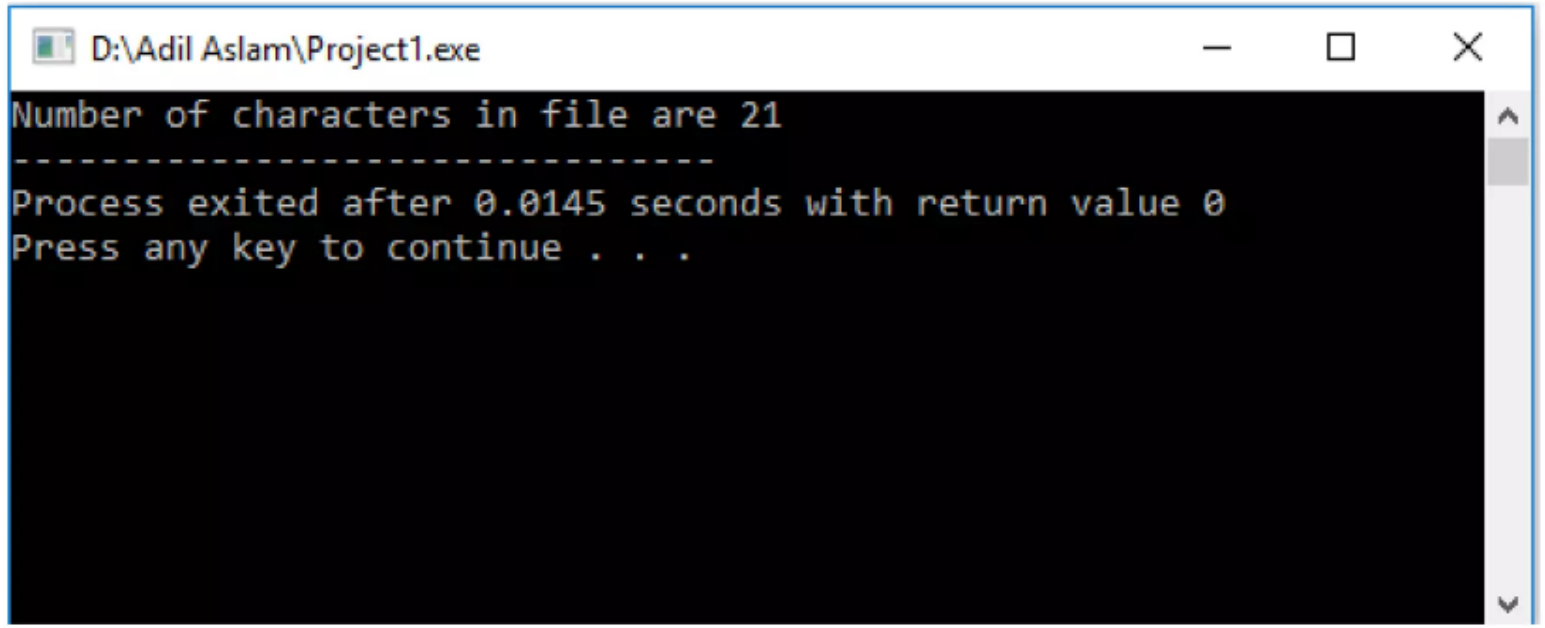
Program to Count Number of Characters-1

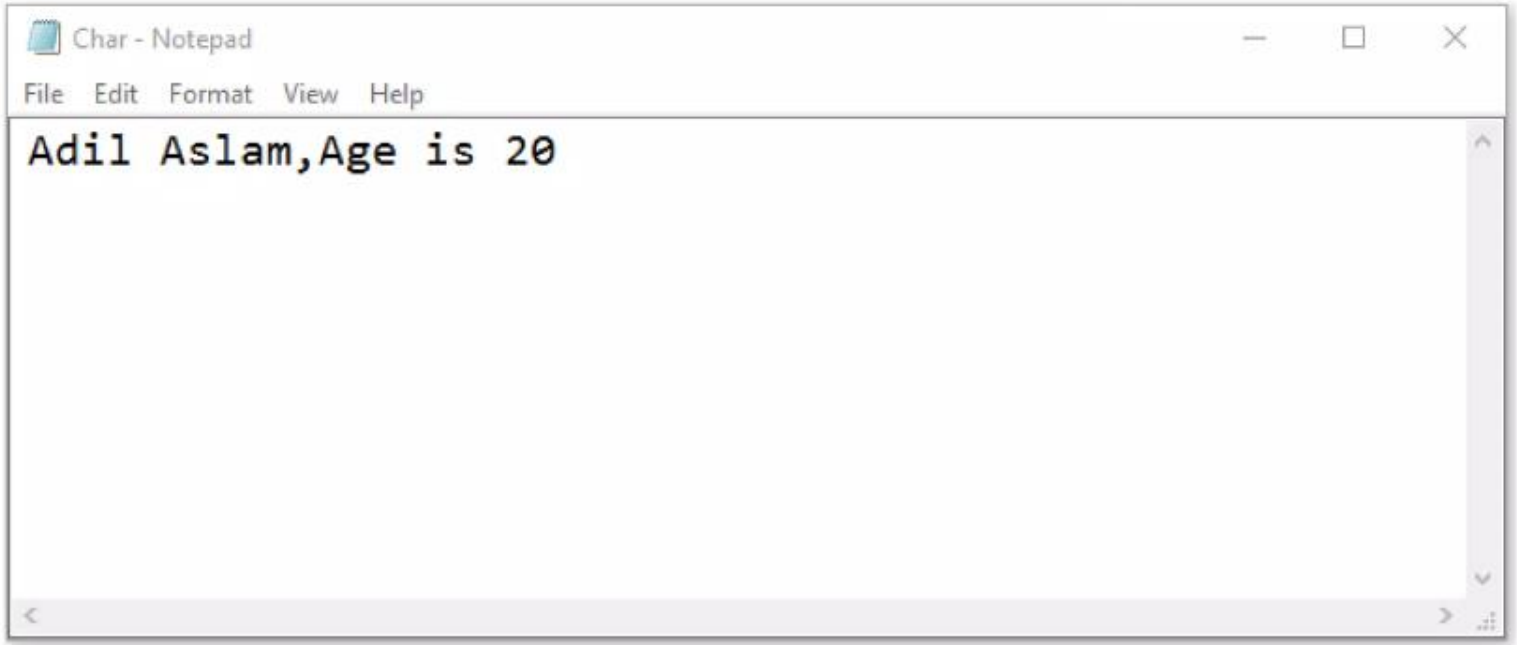
```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
   ifstream fin;
   fin.open("Char.txt");

   int count = 0;
   char ch;
         if (! fin)
         {
                  cout << "Sorry, file can not be opened!" ;
         }
```

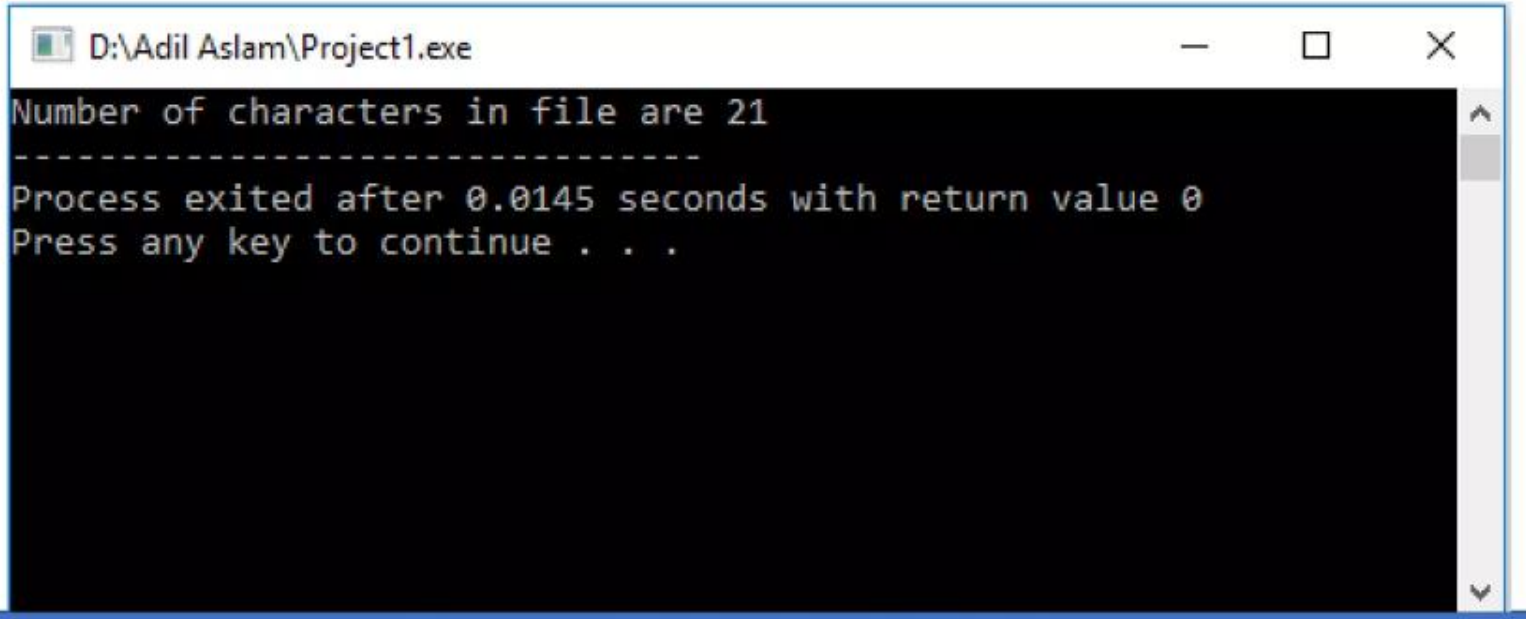
Program to Count Number of Characters-2

```
        else
        {
                while(!fin.eof())
                {
                        fin.get(ch);
                        count++;
                }
        cout << "Number of characters in file are " << count;
    }
    fin.close();
    return 0;
}
```

Output of the Previous Program is :




Char - Notepad
File Edit Format View Help
Adil Aslam,Age is 20

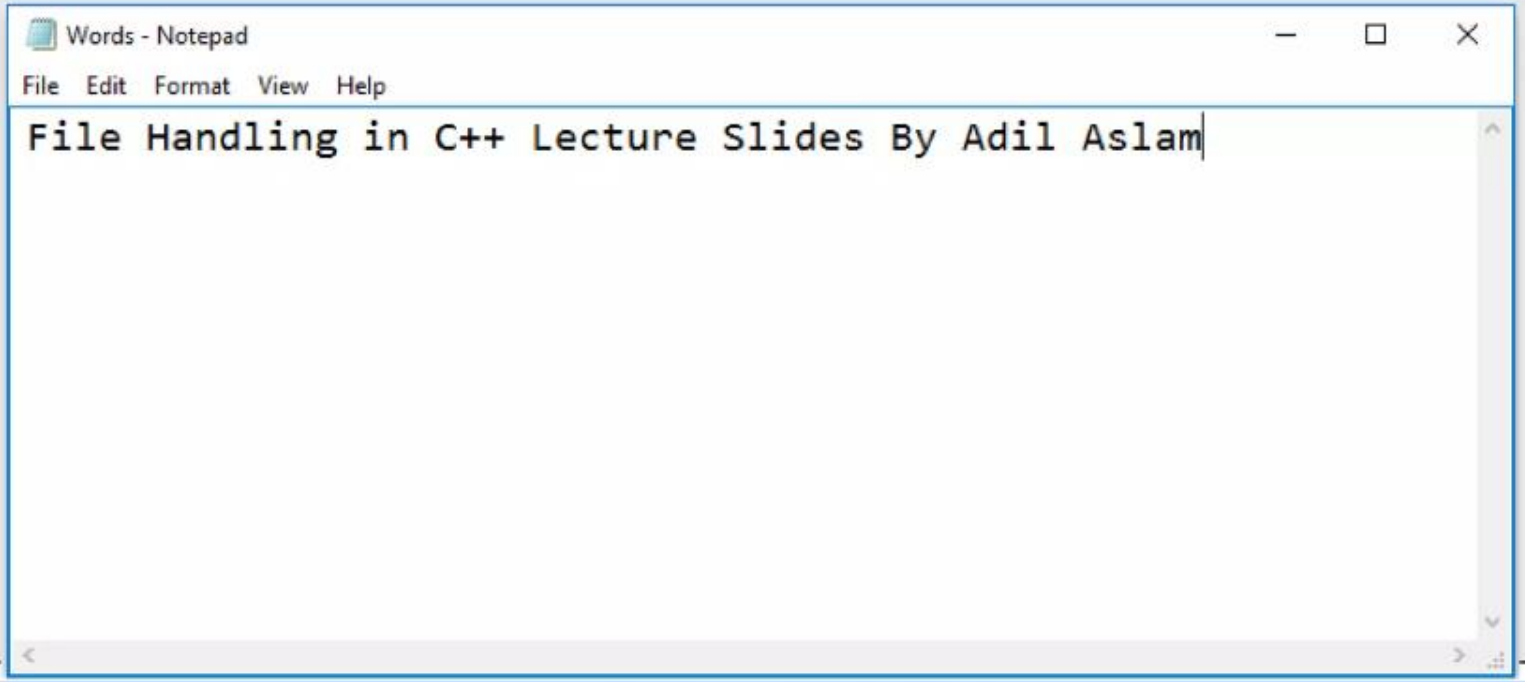

D:\Adil Aslam\Project1.exe
Number of characters in file are 21

Process exited after 0.0145 seconds with return value 0
Press any key to continue . . .

Example No. 5

Read the Following File .
File Name is “Words.txt”.

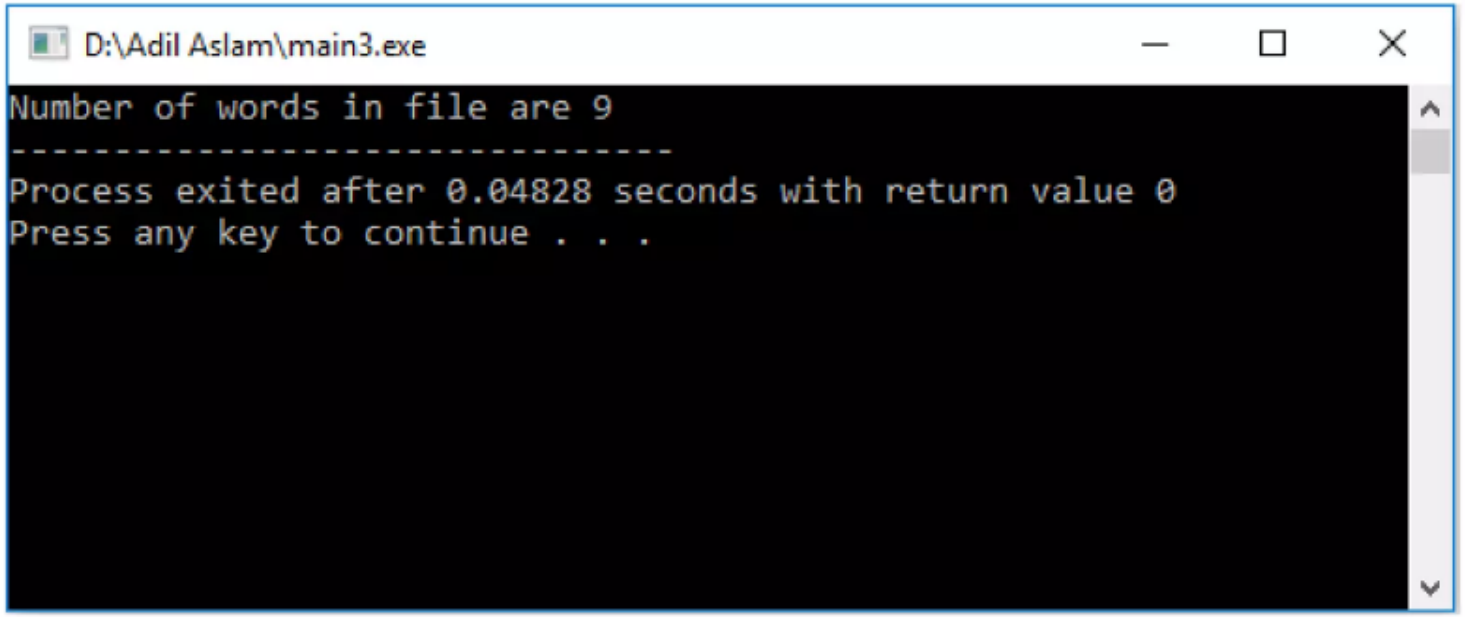
Program To Count Number Of Words



Program to Count Number of Words

```
#include<iostream>
#include<fstream>
using namespace std;
int main() {
   ifstream file;
   file.open("Words.txt");
   int count = 0;
   char word[30];
   while(!file.eof()) {
      file >> word;
      count++;
   }
   cout << "Number of words in file are " << count;
   file.close();
   return 0;
}
```

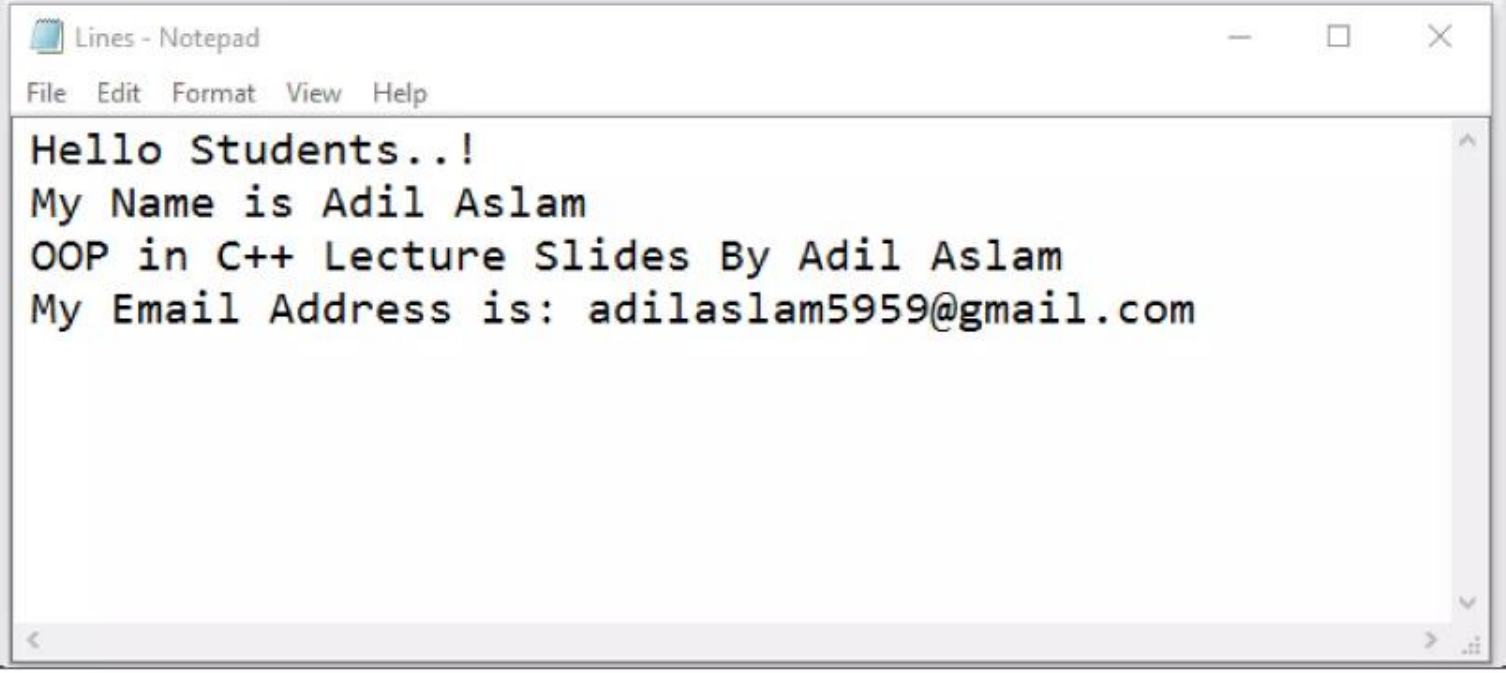
Output of the Previous Program is :



Example No. 5

Read the Following File .
File Name is “Lines.txt”.

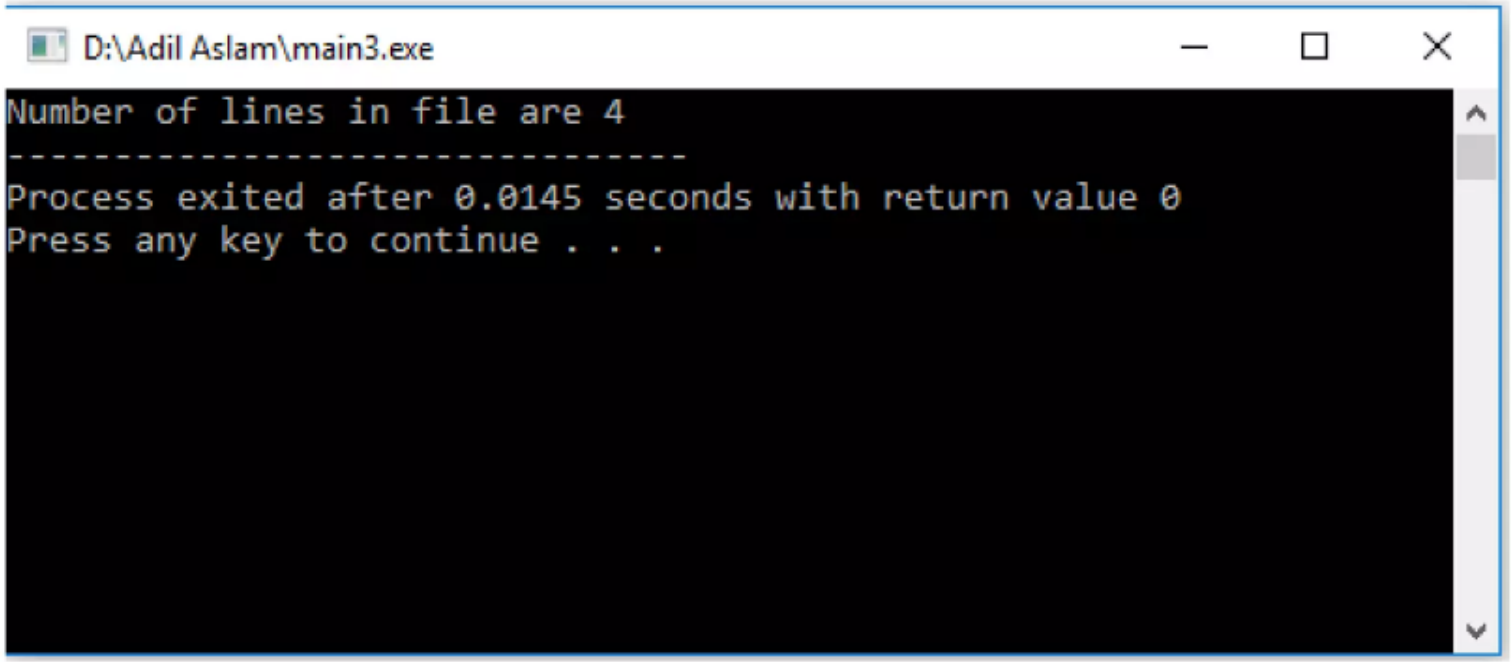
Program To Count Number Of Lines



Program to Count Number of Lines

```
#include<fstream>
#include<iostream>
using namespace std;
int main() {
   ifstream fin;
   fin.open("Lines.txt");
   int count = -1;
   char str[80];
   while(!fin.eof()) {
      fin.getline(str,80);
      count++;
   }
   cout << "Number of lines in file are " << count;
   fin.close();
   return 0;
}
```

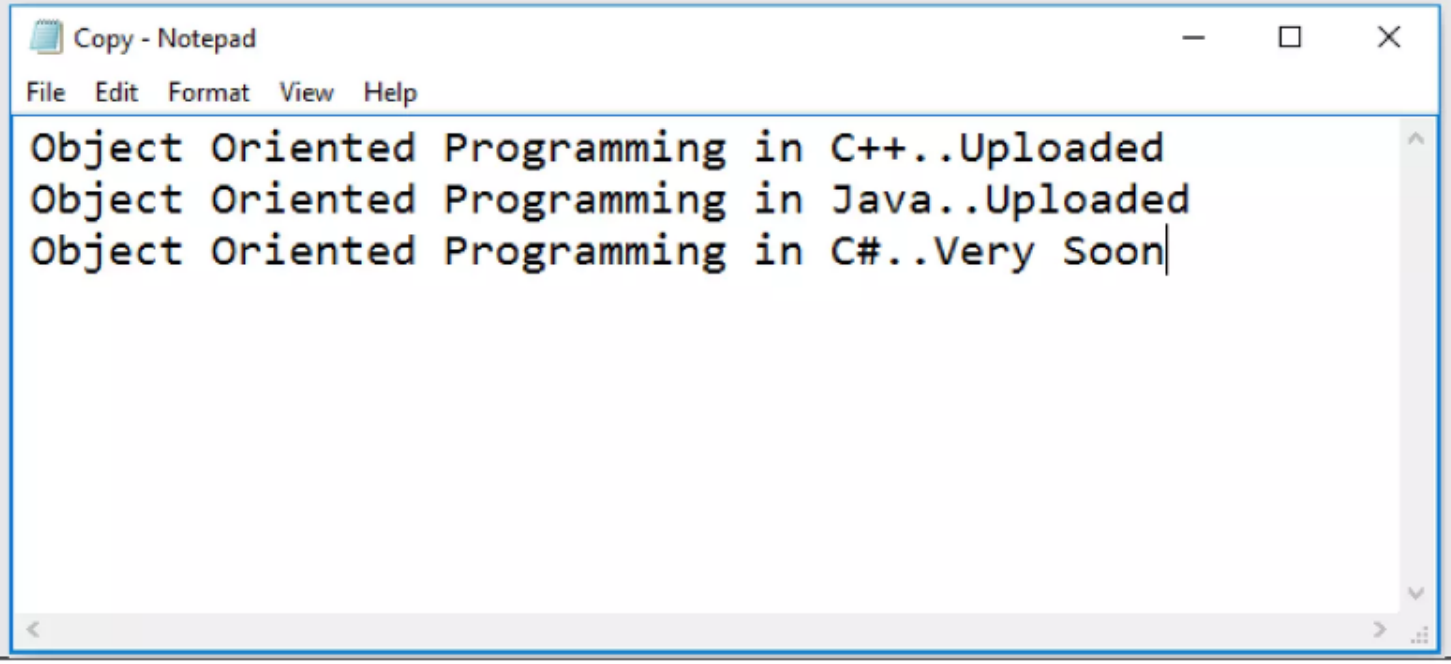
Output of the Previous Program is :



Example No. 5

Read the Following File .
File Name is “Copy.txt”.

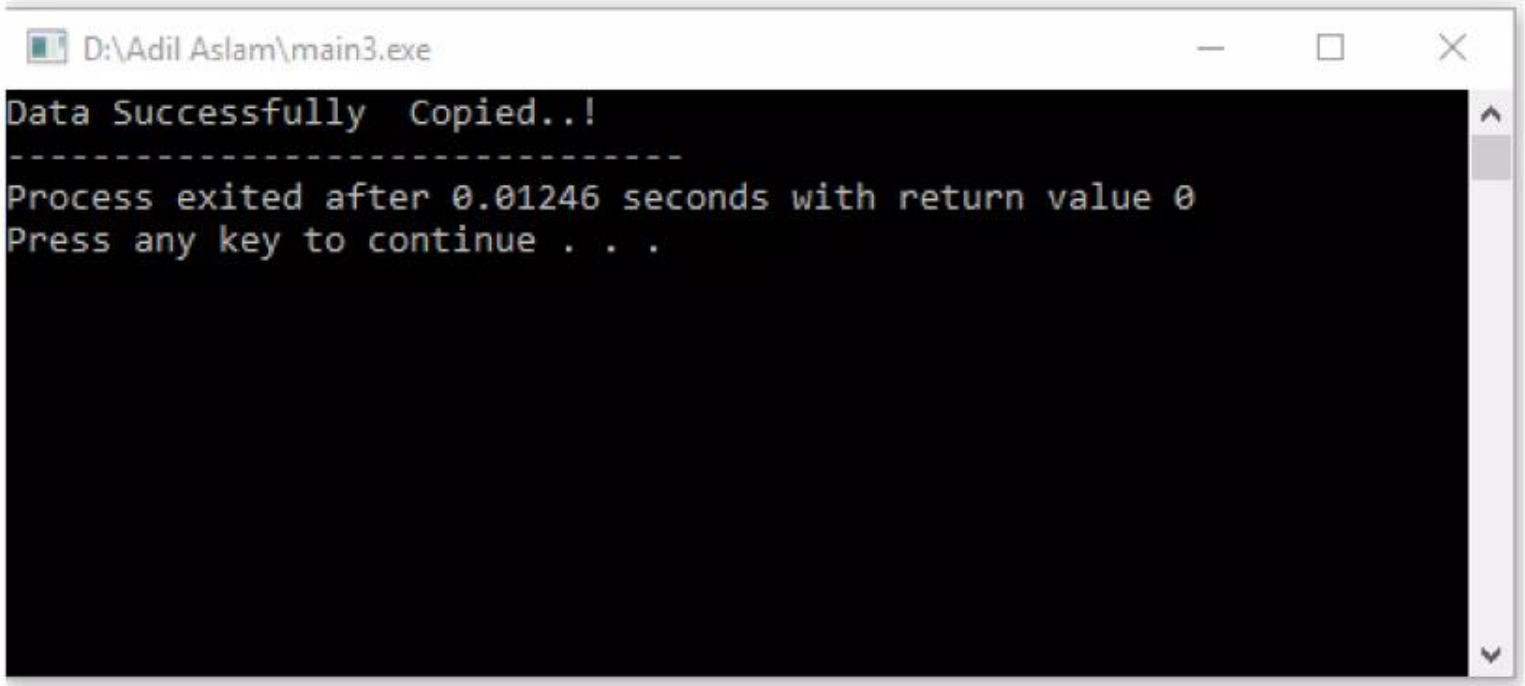
Program To Copy Contents Of File To Another File



Program to Copy Contents of File to Another File

```
#include<iostream>
#include<fstream>
using namespace std;
int main() {
   ifstream fin;
   fin.open("Copy.txt");
   ofstream fout;
   fout.open("sample.txt");
   char ch;
   while(!fin.eof()) {
      fin.get(ch);
      fout << ch;   }
   cout<<"Data Successfully  Copied..!";
   fin.close();
   fout.close();
   return 0;
}
```

Output of the Previous Program is :



Output File Handling

- Several things can be done with output files
  - Create a new file on the disk and write data in it
  - Open an existing file and overwrite it in such a manner that all the old information is lost from it and new information is stored
  - Open an existing file and append it in at the end
  - Open an existing file and modify in it in such a way that it can be written anywhere in the file

File Opening Mode

- **The syntax of open function is:**

```
handler.open(fileName, mode)
```

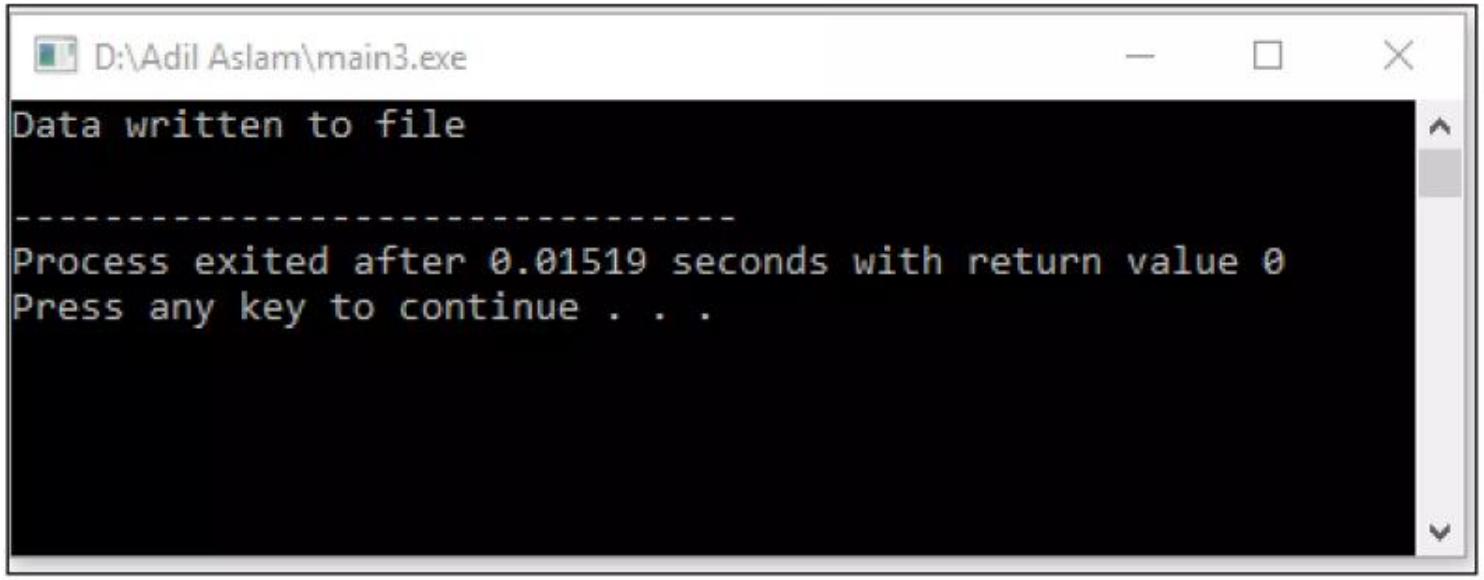
- **Example:**

```
ofstream myFile;
myFile.open("testfile.txt" , ios::out);
```

Open a File for Writing

```
#include<iostream>
#include<fstream>
using namespace std;
int main() {
ofstream ofile;            // declaring an object of class
ofile.open("Myfile.txt");  // open "file.txt" for writing data
/* write to a file */
ofile << "This is a line in a file" << endl;
ofile << "This is another line" << endl;
  /* write to a console */
cout << "Data written to file" << endl;
ofile.close();     // close the file
return 0;
}
```

Output of Previous Program is



For now open your project saving location, open project folder then you can see a text file with name of Myfile

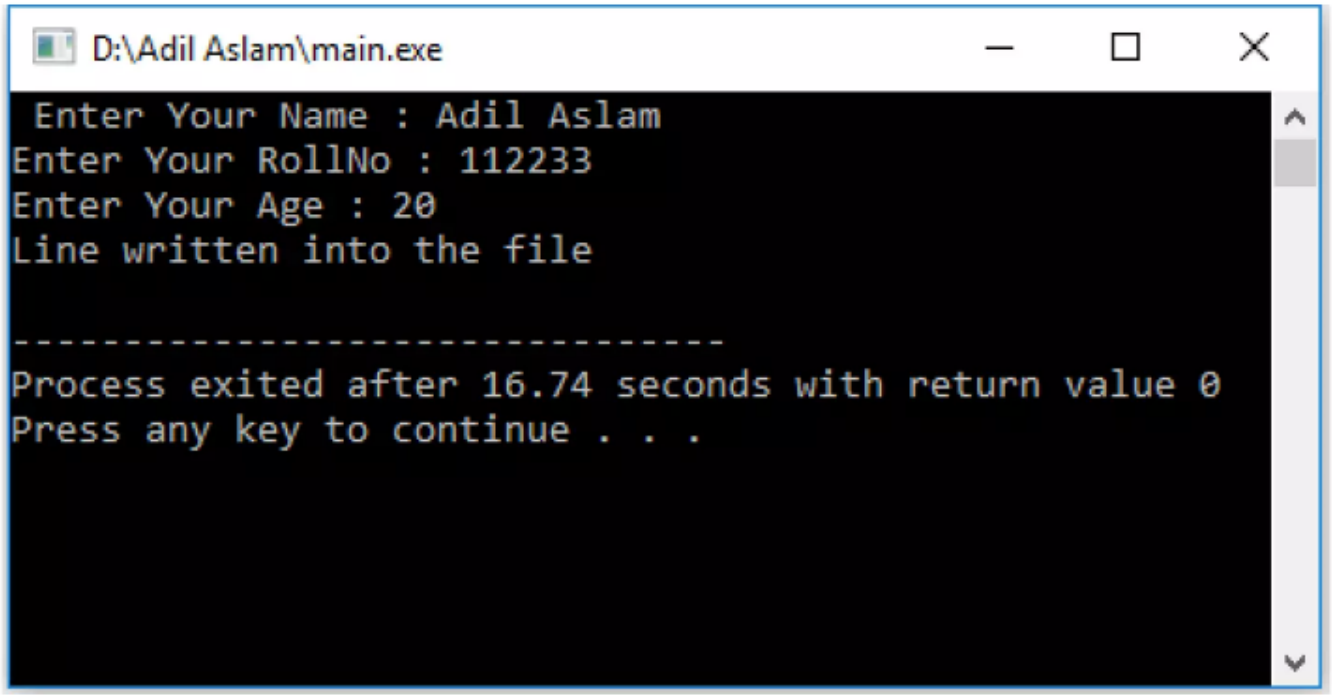
Open a file and Append data to the end of the File-1

```
#include<iostream>
#include<fstream>
using namespace std;
int main() {
  char line[100];
  fstream file;    // declare an object of fstream class
  // open file in append mode
  file.open("file.txt", ios :: out | ios :: app);
  if (file.fail()) {    // check if file is opened successfully
    // file opening failed
    cout << "Error Opening file ... " << endl;
  }
```

Open a file and Append data to the end of the File-2

```
else {
    // proceed with further operations
    cout << " Enter Your Name : ";
    cin.getline(line, 100);
    file << line << endl;     // Append the line to the file
    cout << "Enter Your RollNo : ";
    cin.getline(line, 100);
    file << line << endl;
    cout << "Enter Your Age : ";
    cin.getline(line, 100);
    file << line << endl;
    cout << "Line written into the file" << endl;
    }
  return 0;
}
```

Output of the Previous Program is



Types of Files

- Types of file supported by C++:
- Text Files
- Binary Files

- A **text file** consists of readable characters separated into lines by newline characters. (On most PCs, the newline character is actually represented by the two-character sequence of carriage return (ASCII 13), line feed (ASCII 10).

Types of Files

- A **binary file** stores data to disk in the same form in which it is represented in main memory.
- If you ever try to edit a binary file containing numbers you will see that the numbers appear as nonsense characters. Not having to translate numbers into a readable form makes binary files somewhat more efficient.
- Binary files also do not normally use anything to separate the data into lines. Such a file is just a stream of data with nothing in particular to separate components.

Types of Files

- When using a binary file we write whole record data to the file at once. When using a text file, we write out separately each of the pieces of data about a given record.
- The text file will be readable by an editor, but the numbers in the binary file will not be readable in this way.
- The programs to create the data files will differ in how they open the file and in how they write to the file.

Types of Files

- For the **binary file** we will use **`write`** to write to the file, whereas for the text file we will use the usual output operator(<<) and will output each of the pieces of the record separately.

- With the **binary file** we will use the **`read`** function to read a whole record, but with the text file we will read each of the pieces of record from the file separately, using the usual input operator(>>)

Difference Between Test and Binary File

- Text file is human readable because everything is stored in terms of text. In binary file everything is written in terms of 0 and 1, therefore binary file is not human readable.
- A newline(\n) character is converted into the carriage return-linefeed combination before being written to the disk. In binary file, these conversions will not take place.
- In text file, a special character, whose ASCII value is 26, is inserted after the last character in the file to mark the end of file. There is no such special character present in the binary mode files to mark the end of file.
- In text file, the text and characters are stored one character per byte. For example, the integer value 23718 will occupy 2 bytes in memory but it will occupy 5 bytes in text file. In binary file, the integer value 23718 will occupy 2 bytes in memory as well as in file.

File Handling

- C++'s standard library called **fstream**, defines the following classes to support file handling.

- **ofstream class :** Provides methods for writing data into file. Such as, open(), put(), write(), seekp(), tellp(), close(), etc.
- **ifstream class :** Provides methods for reading data from file. Such as, open(),get(), read(), seekg(), tellg(), close(), etc.
- **fstream class :** Provides methods for both writing and reading data from file. The fstream class includes all the methods of ifstream and ofstream class.

Read Write Object using read() and write() Function

- There are member functions **read( )** and **write( )** in the **fstream** class which allows reading and writing of class objects.
- These functions can also be used to write array elements into the file.
- The write() function is used to write object or record (sequence of bytes) to the file. A record may be an array, structure or class.
- The read() function is used to read object (sequence of bytes) to the file.

Read Write Object using read() and write() Function

Syntax of write() function

```
stream fout;
fout.write( (char *) &obj, sizeof(obj) );
```

- The write() function takes two arguments.
- **&obj :** Initial byte of an object stored in memory.
- **sizeof(obj) :** size of object represents the total number of bytes to be written from initial byte.

Read Write Object using read() and write() Function

Syntax of read() function

```
fstream fin;
fin.read( (char *) &obj, sizeof(obj) );
```

- The read() function takes two arguments.
- **&obj :** Initial byte of an object stored in file.
- **sizeof(obj) :** size of object represents the total number of bytes to be read from initial byte.

- The read() function returns NULL if no data read.

Example of write() Function-1

```
#include<fstream>
#include<iostream>
#include<conio.h>
using namespace std;
class Student {
        int roll;
  char name[25];
  float marks;
   void getdata()
  {
     cout<<"\n\nEnter Roll : ";
     cin>>roll;
     cout<<"\nEnter Name : ";
     cin>>name;
     cout<<"\nEnter Marks : ";
     cin>>marks;
  }
```

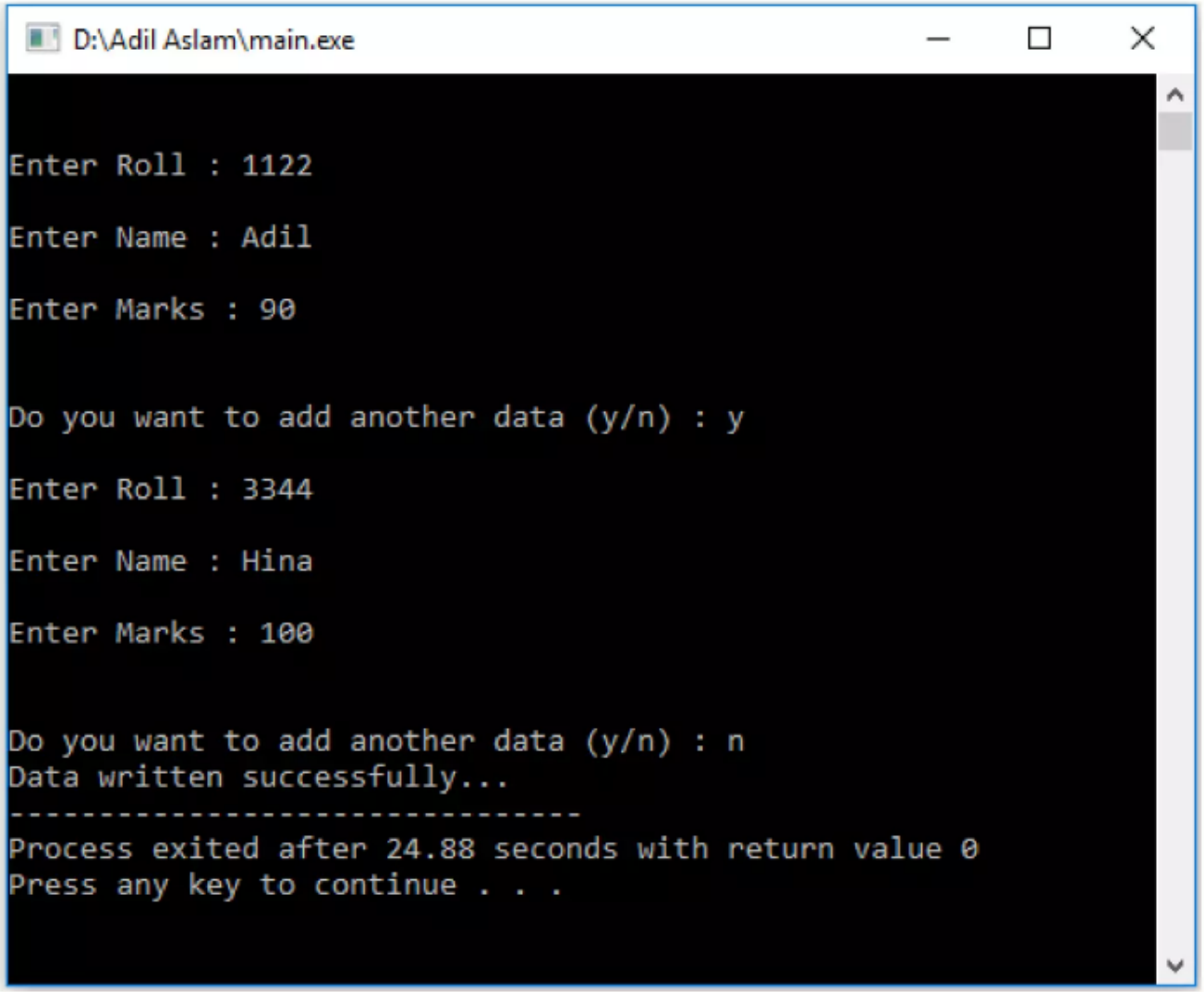
Example of write() Function-2

```
public:
    void AddRecord()
   {
                    fstream f;
                   Student Stu;
                    f.open("Student.dat",ios::app|ios::binary);
                    Stu.getdata();
                    f.write( (char *) &Stu, sizeof(Stu) );
                    f.close();

   }
};
```

Example of write() Function-3

```
int main()
{


   Student S;
   char ch='n';
  do
   {
          S.AddRecord();
          cout<<"\n\nDo you want to add another data (y/n) : ";
              ch = getche();
   } while(ch=='y' || ch=='Y');


   cout<<"\nData written successfully...";
}
```

Output of the Previous Program is :



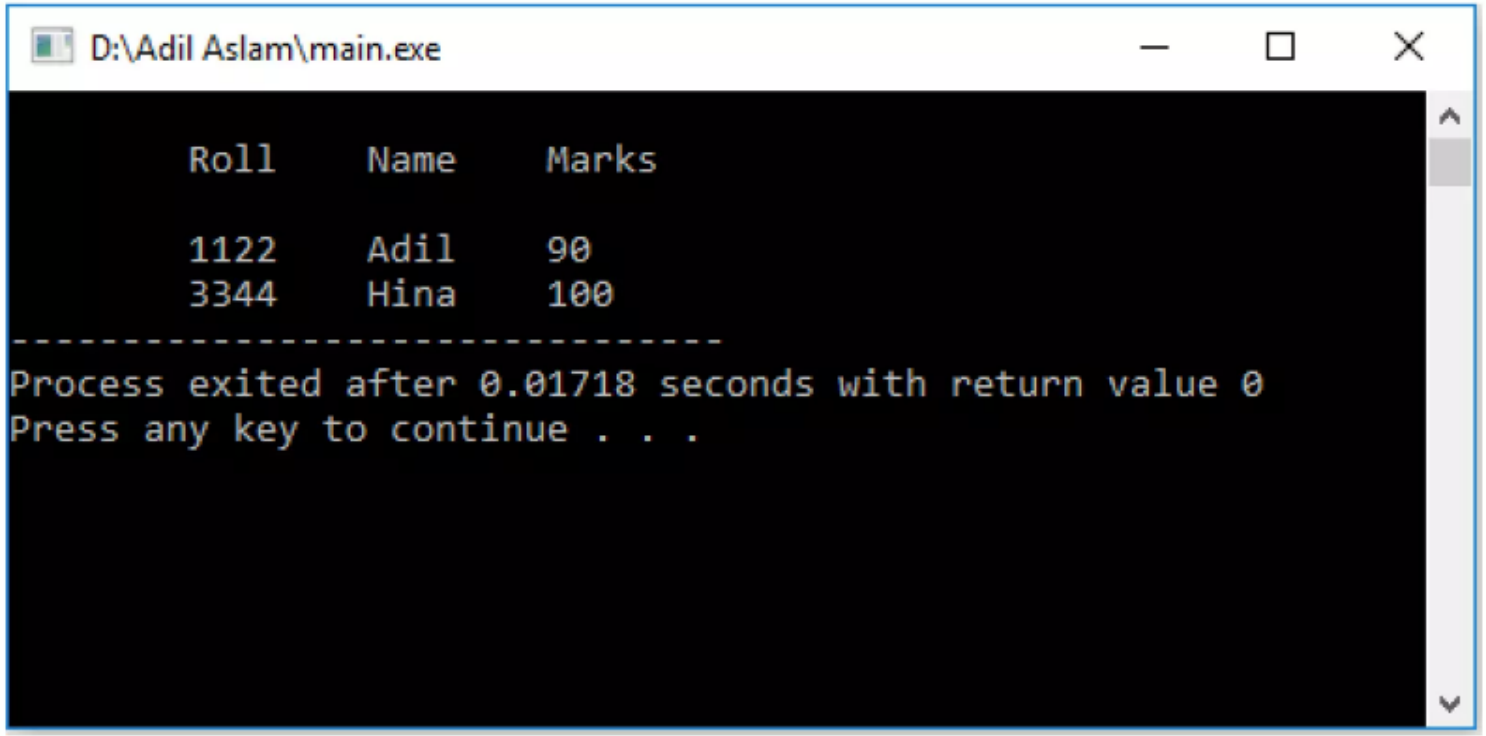
Example of read() Function-1

```
#include<fstream>
#include<iostream>
#include<conio.h>
using namespace std;
class Student
{

   int roll;
   char name[25];
   float marks;
         void putdata()
         {
               cout<<"\n\t"<<roll<<"\t"<<name<<"\t"<<marks;
   }
```

Example of read() Function-2

```
public:
        void Display() {
        fstream f;
        Student Stu;
        f.open("Student.dat",ios::in|ios::binary);
        cout<<"\n\tRoll\tName\tMarks\n";
        while( (f.read((char*)&Stu,sizeof(Stu))) != NULL )
            Stu.putdata();
        f.close();
    }
};
int main() {
        Student S;
        S.Display();
}
```

Output of the Previous Program is :



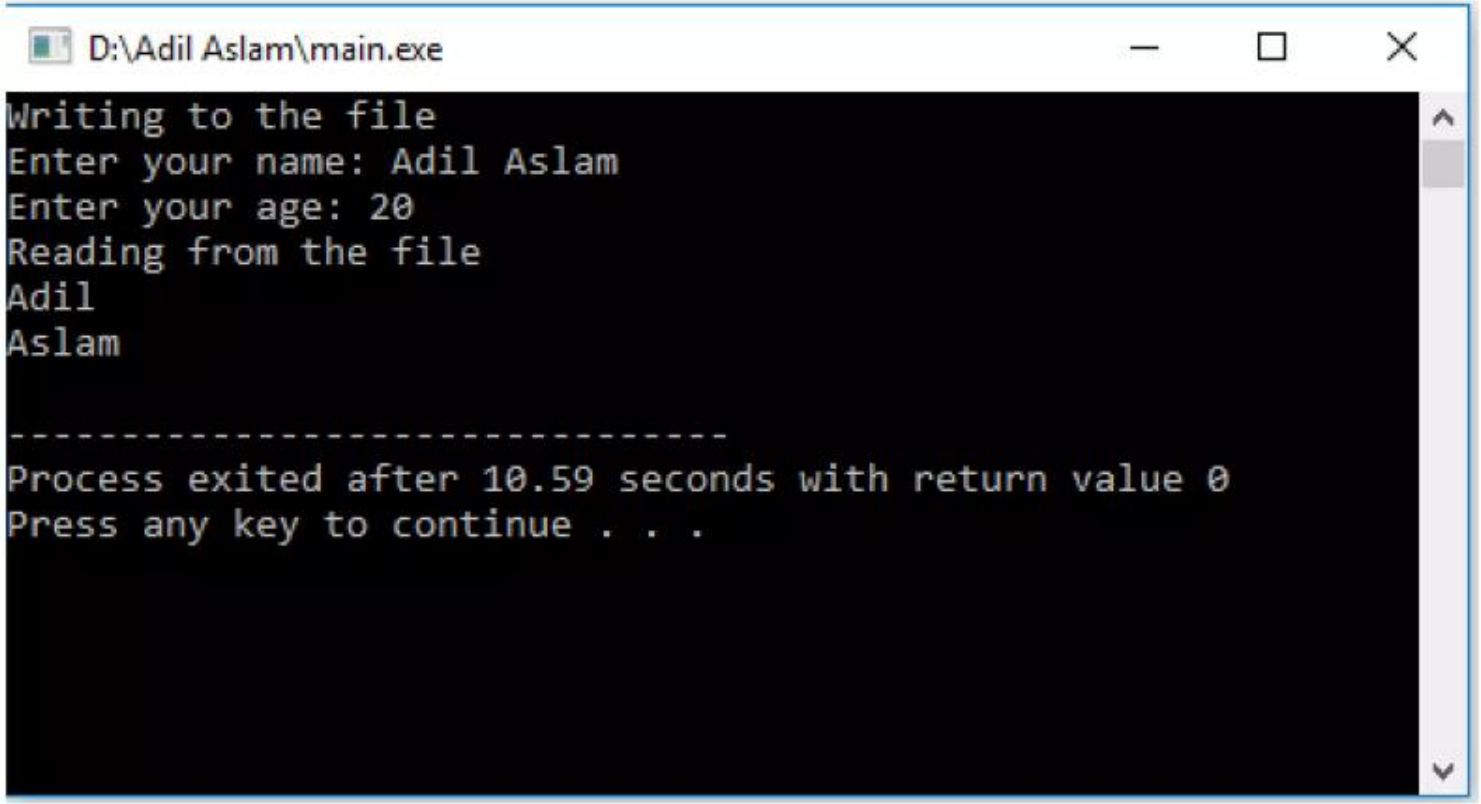
Example of Read & Write-1

```
#include <fstream>
#include <iostream>
using namespace std;
int main () {
  char data[100];
  // open a file in write mode.
  ofstream outfile;
  outfile.open("afile.dat");
  cout << "Writing to the file" << endl;
  cout << "Enter your name: ";
  cin.getline(data, 100);
  // write inputted data into the file.
  outfile << data << endl;
  cout << "Enter your age: ";
  cin >> data;
  cin.ignore();
```

Example of Read & Write-2

```
// again write inputted data into the file.
  outfile << data << endl;
   outfile.close();
  // open a file in read mode.
  ifstream infile;
  infile.open("afile.dat");
  cout << "Reading from the file" << endl;
  infile >> data;
  // write the data at the screen.
  cout << data << endl;
  // again read the data from the file and display it.
  infile >> data;
  cout << data << endl;
  // close the opened file.
  infile.close();
  return 0;
}
```

Output of the Previous Program is :



Writing Class Objects to a File-1

```
#include<iostream>
#include<fstream>
using namespace std;

// define a class to store student data
class student {
        int roll;
        char name[30];
        float marks;
public:
        student() { }
        void getData();    // get student data from user
        void displayData();    // display data
};
```

Writing Class Objects to a File-2

```
void student :: getData() {
  cout << "\n Enter Roll No. : ";
  cin >> roll;
  // ignore the newline char inserted when you press enter
  cin.ignore();
  cout << "Enter Name : ";
  cin.getline(name, 30);
  cout << "Enter Marks : ";
  cin >> marks;
}
void student :: displayData() {
  cout << "\n Roll No. : " << roll << endl;
  cout << "Name : " << name << endl;
  cout << "Marks : " << marks << endl;
}
```

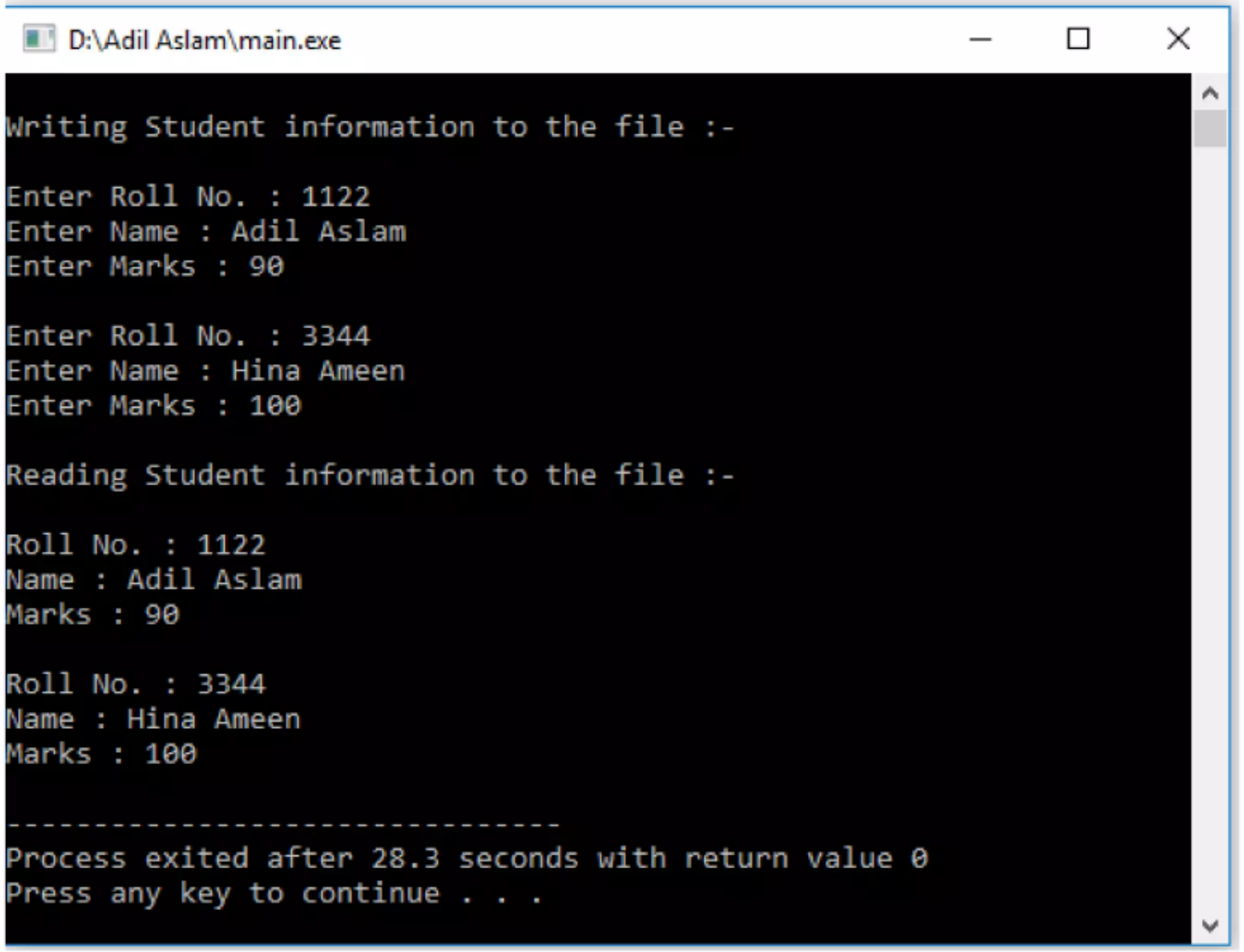
Writing Class Objects to a File-3

```
int main() {
  student s[2];     // array of 3 student objects
  fstream file;
  int i;
   file.open("objects.txt", ios :: out);    // open file for writing
  cout << "\nWriting Student information to the file :- " << endl;
  for (i = 0; i < 2; i++) {
    s[i].getData();
    // write the object to a file
    file.write((char *)&s[i], sizeof(s[i]));
  }
```

Writing Class Objects to a File-4

```
file.close();   // close the file
 file.open("objects.txt", ios :: in);  // open file for reading
  cout << "\n Reading Student information to the file :- " <<
endl;
  for (i = 0; i < 2; i++) {
    // read an object from a file
    file.read((char *)&s[i], sizeof(s[i]));
    s[i].displayData();
  }
  file.close();    // close the file
  return 0;
}
```

Output of the Previous Program is :



Example No. 6

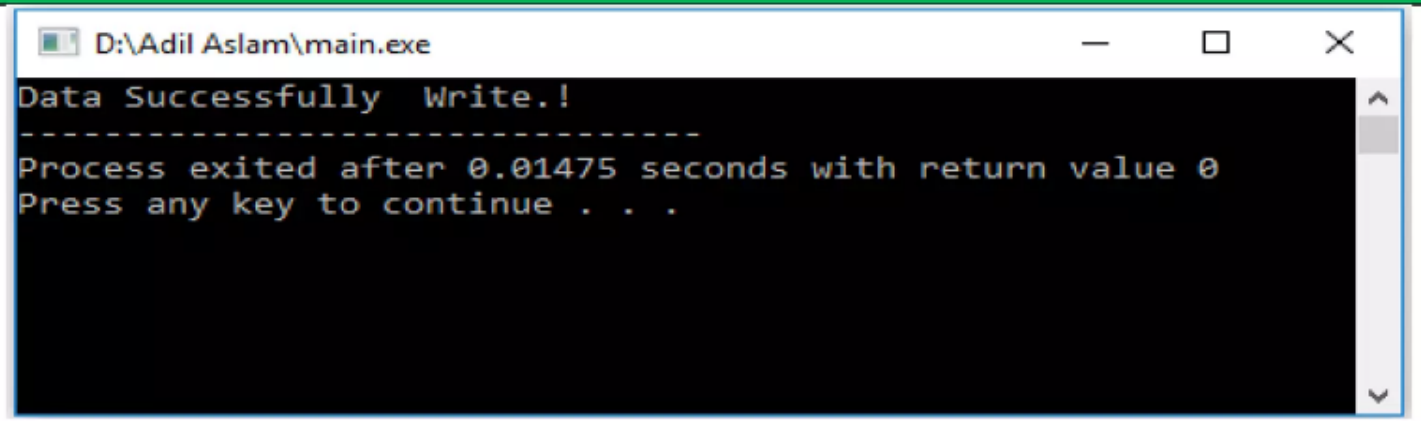
Write the Following File .
File Name is “Numbers.txt”.

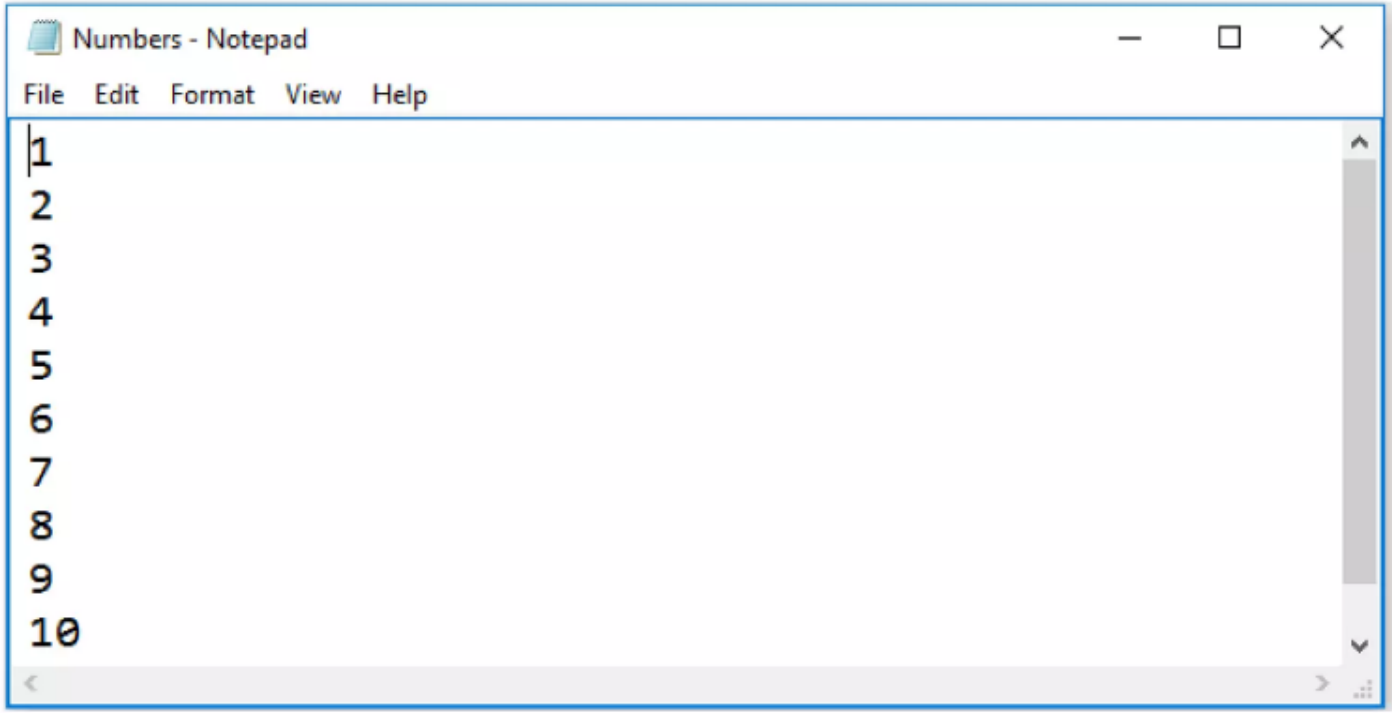
C++ Program to Write Number 1 to 10 in a Data file

Write Number 1 to 10 in a Data file

```
#include<fstream>
#include<iostream>
using namespace std;
int main() {
        ofstream out;
        if (! out) {
                cout << "Sorry, file can not be open!!!" << endl;
        }
        else {
        out.open("Numbers.TXT");
        for(int i=1;i<=10;i++)
                out<<i<<endl;
        out.close();
  }
cout<<"Data Successfully  Write.!";
        return 0;
}
```

Output of the Previous Program is :





Example No. 7

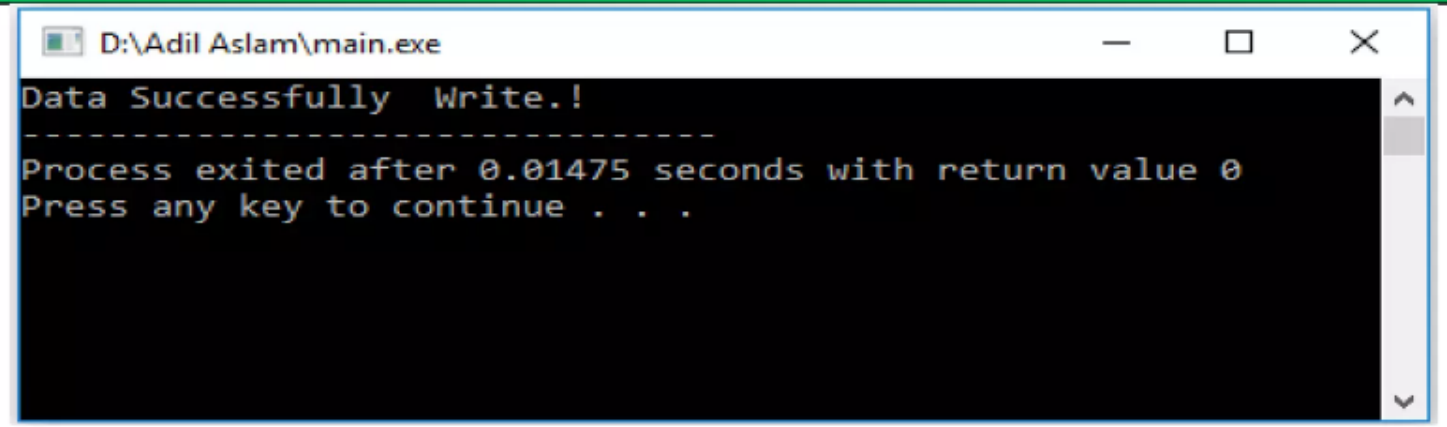
Write the Following File .
File Name is “String.txt”.

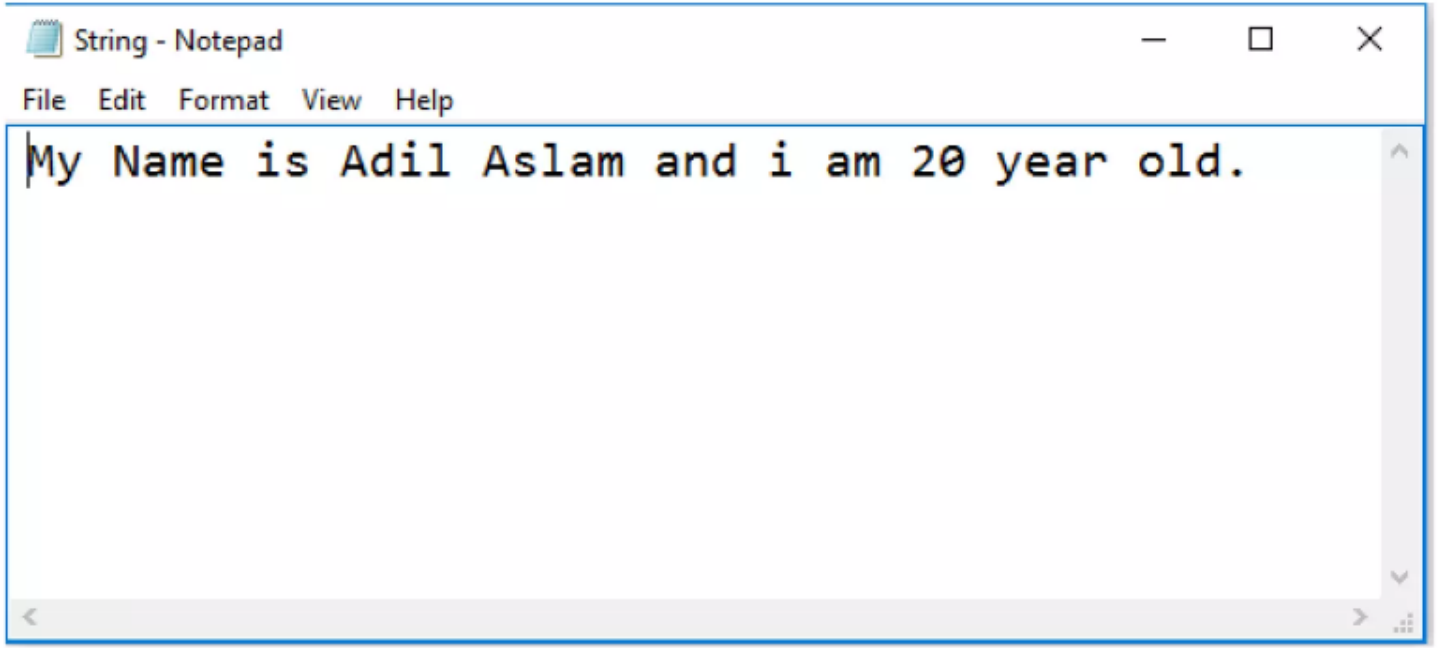
C++ Program, which Initializes a String Variable and Outputs the String to the Disk File

Initializes a String Variable and Outputs the String

```
#include<fstream>
#include<iostream>
using namespace std;
int main() {
        ofstream fout;
        if (! fout) {
                cout << "Sorry, file can not be open!!!" << endl;   }
        else {
        fout.open("String.txt");
        char str[300]="My Name is Adil Aslam and i am 20 year old.";
        fout<<str;
        fout.close();
  }
cout<<"Data Successfully  Write.!";
        return 0;
}
```

Output of the Previous Program is :





Example No. 7

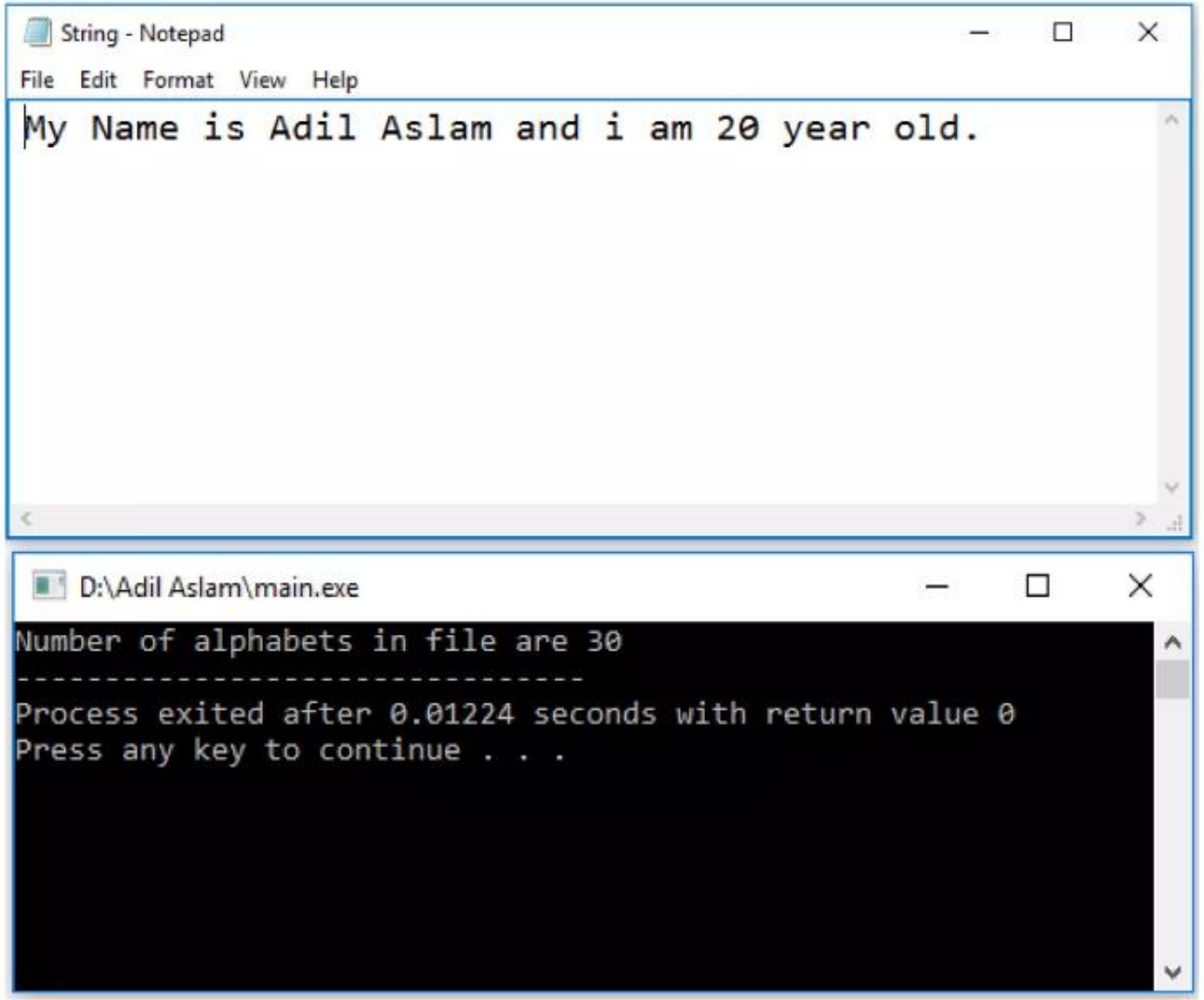
Read the Following File .
File Name is “String.txt”.

User-defined Function in C++ to read the Content from a text file *String.txt*, count and Display the Number of Alphabets Present in it

Count Number of Alphabets

```
void alphabets() {
        ifstream fin;
        fin.open("String.txt");
        char ch;
        int count=0;
        while(!fin.eof())
        {
                fin.get(ch);
                if(isalpha(ch))
                        count++;  }
        cout<<"Number of alphabets in file are "<<count;
        fin.close();
}
int main()
 {
        alphabets();
 }
```

Output of the Previous Program is :



Example No. 8

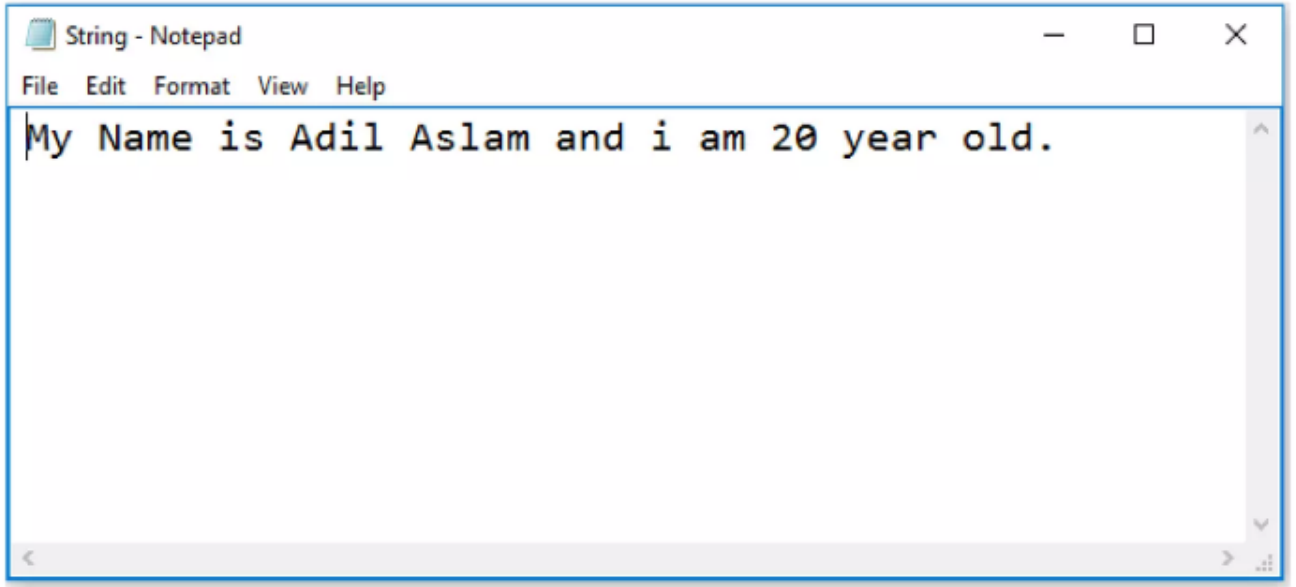
Read the Following File .
File Name is “String.txt”.

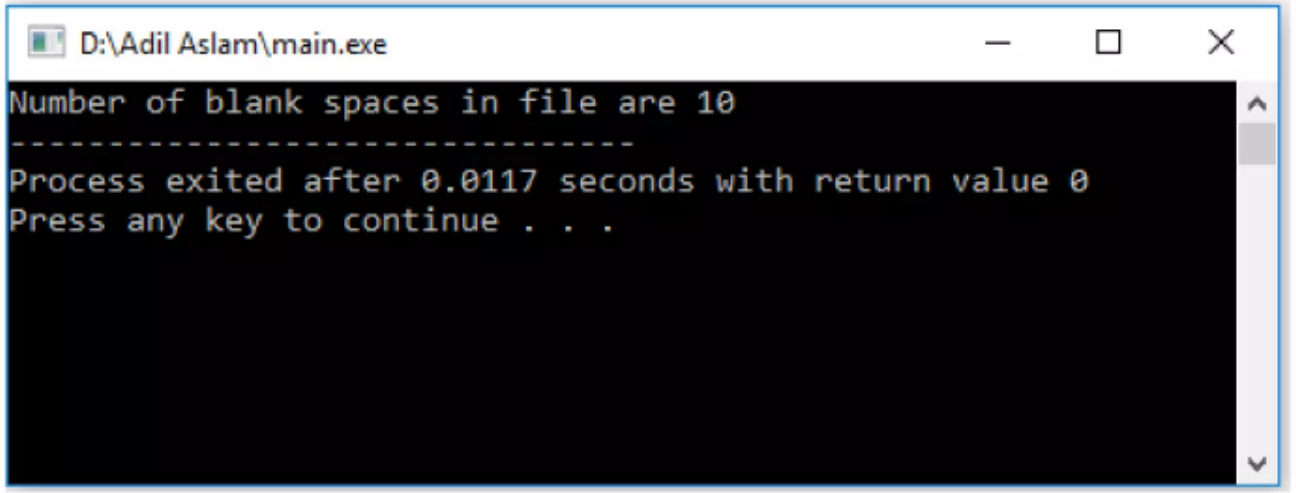
User Defined Function in C++ to Count the Number of blank Present in a text file Named "*String.txt* ".

Count the Number of blank

```
void blankspace() {
        ifstream fin;
        fin.open("String.txt");
        char ch;
        int count=0;
        while(!fin.eof()) {
                fin.get(ch);
                if(ch==' ')
                        count++;
        }
        cout<<"Number of blank spaces in file are "<<count;
        fin.close();
}
int main()  {
        blankspace();
 }
```

Output of the Previous Program is :





Example No. 9

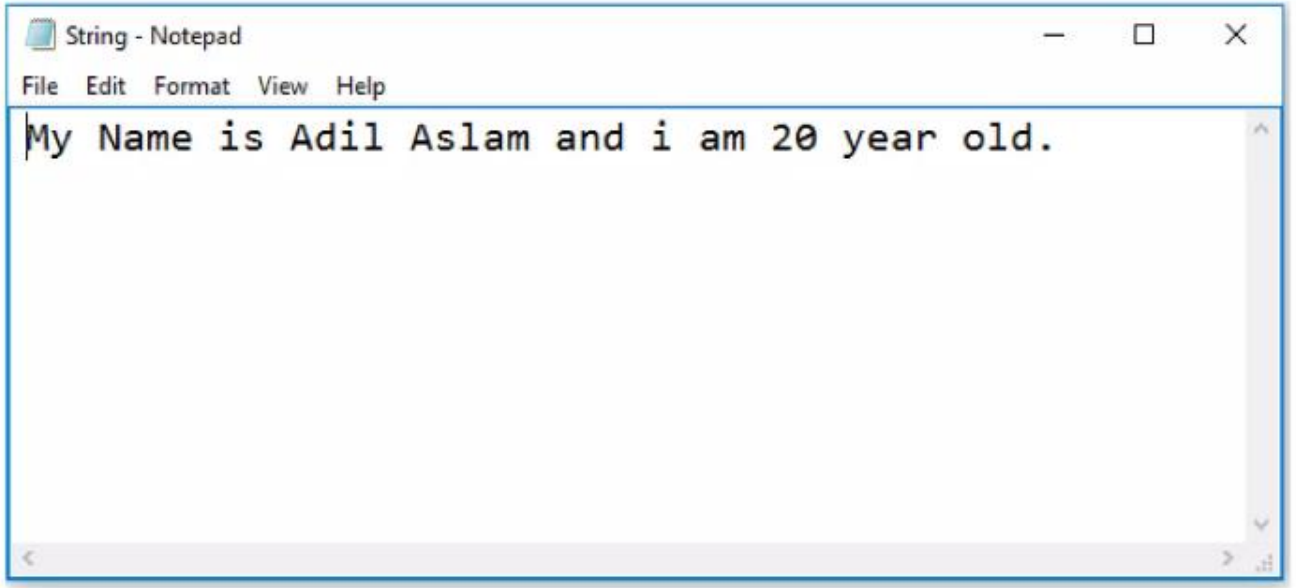
Read the Following File .
File Name is “String.txt”.

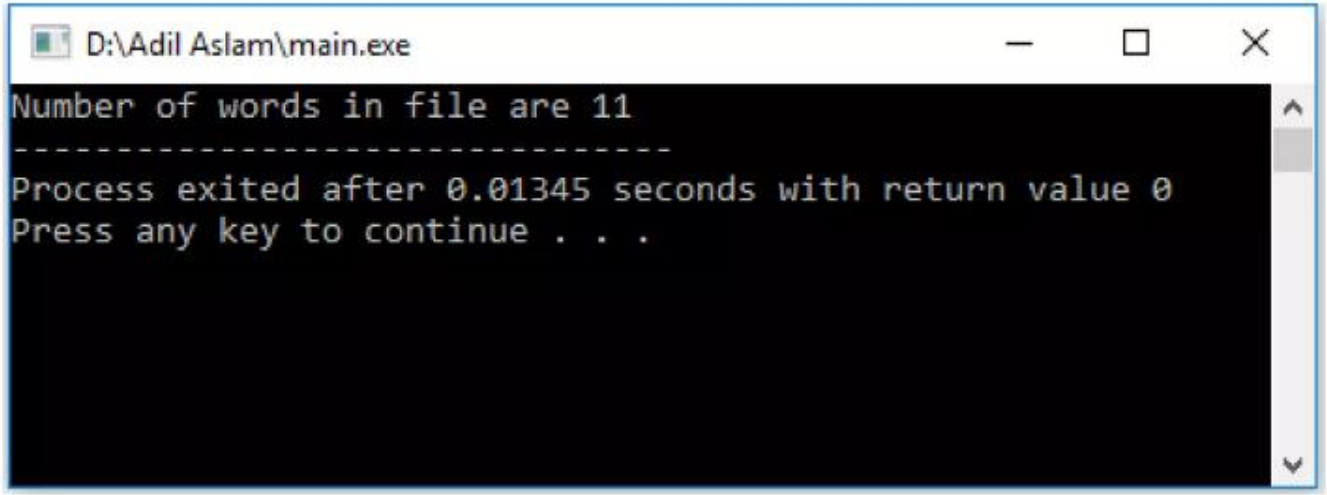
User Defined Function in C++ to Count Number of Words in a text file named "*String.txt* "

Count Number of Words

```
void countwords() {
        ifstream fin;
        fin.open("String.txt");
        char word[30];
        int count=0;
        while(!fin.eof()) {
                fin>>word;
                count++;
        }
        cout<<"Number of words in file are "<<count;
        fin.close();
}
int main() {
        countwords();
 }
```

Output of the Previous Program is :





Example No. 10

Read the Following File .
File Name is “String.txt”.

User Defined Function in C++ to Print the Count of word the as an Independent word in a text file *“String.txt”*.

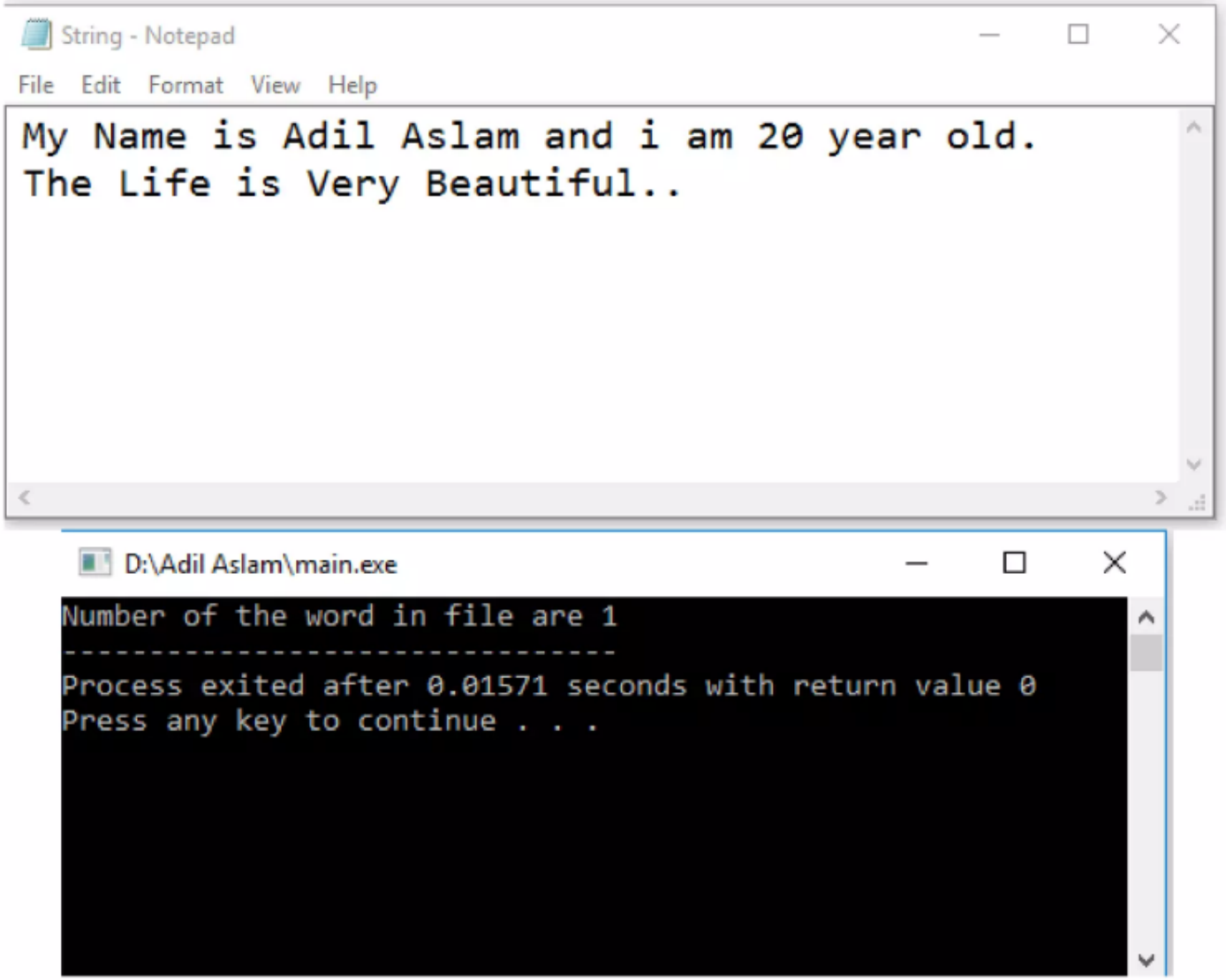
Count of word the as an Independent word-1

```
#include<fstream>
#include<iostream>
#include<cstring>
using namespace std;
void countword() {
        ifstream fin;
        fin.open("String.TXT");
        char word[30];
        int count=0;
```

Count of word the as an Independent word-2

```
        while(!fin.eof())
        {
                fin>>word;
                if(strcmpi(word,"the")==0)
                        count++;
        }
        cout<<"Number of the word in file are "<<count;
        fin.close();
}
int main() {
        countword();
 }
```

Output of the Previous Program is :



Example No. 11

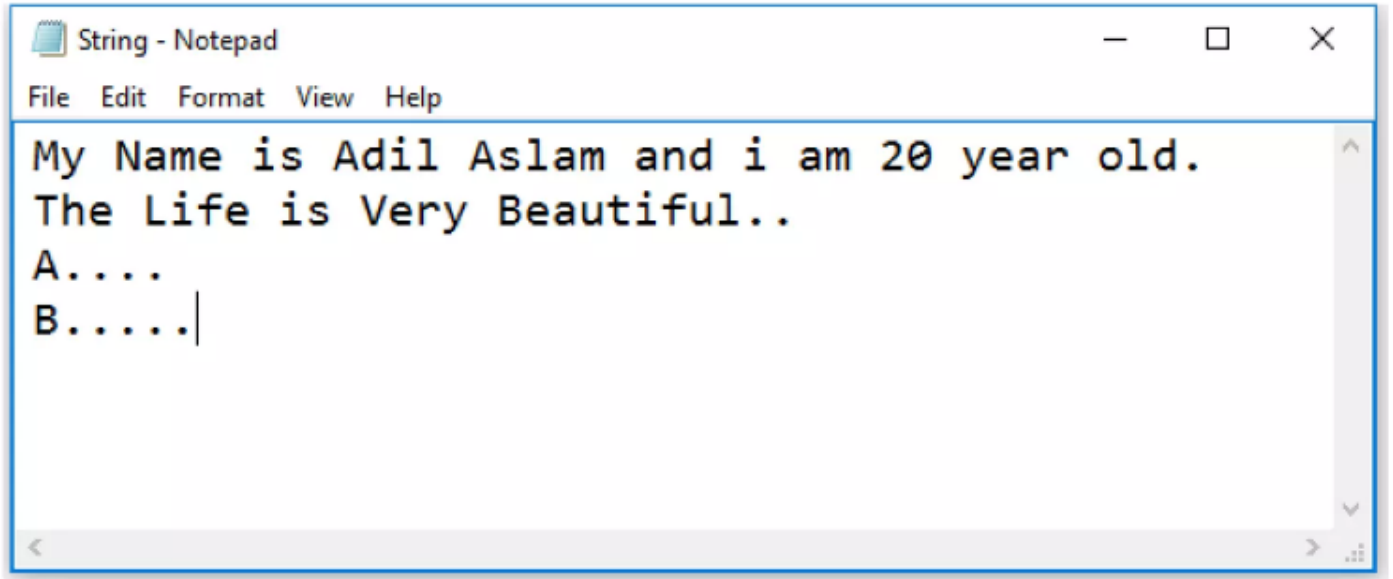
Read the Following File .
File Name is “String.txt”.

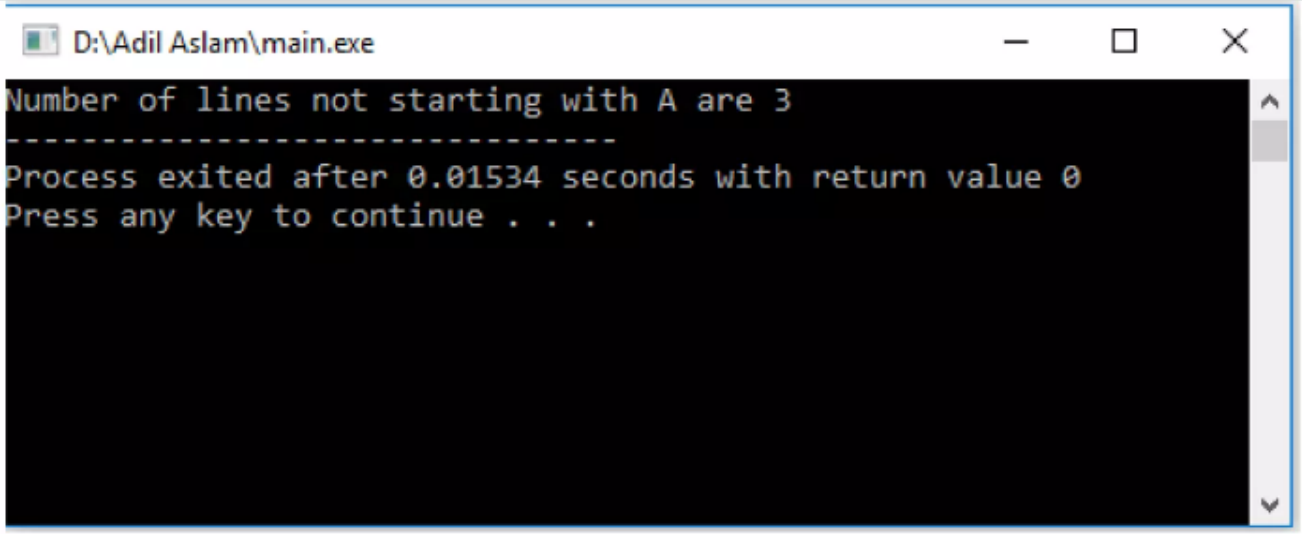
Function in C++ to Count and Display the Number of lines not Starting with Alphabet 'A' Present in a text file "*String.txt*"

Count Number of lines not Starting with Alphabet 'A'

```
void countlines() {
        ifstream fin;
        fin.open("String.TXT");
        char str[80];
        int count=0;
        while(!fin.eof()) {
                fin.getline(str,80);
                if(str[0]!='A')
                        count++;
        }
        cout<<"Number of lines not starting with A are "<<count;
        fin.close();
}
int main() {
        countlines();
}
```

Output of the Previous Program is :





Example No. 12

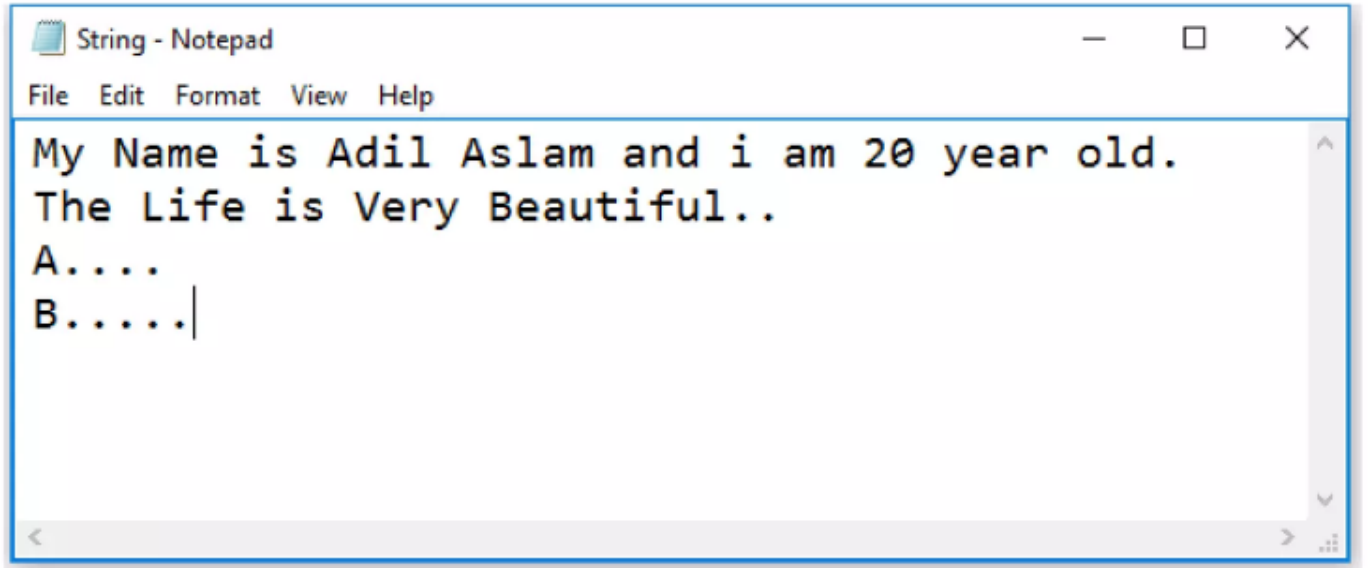
Read the Following File .
File Name is “String.txt”.

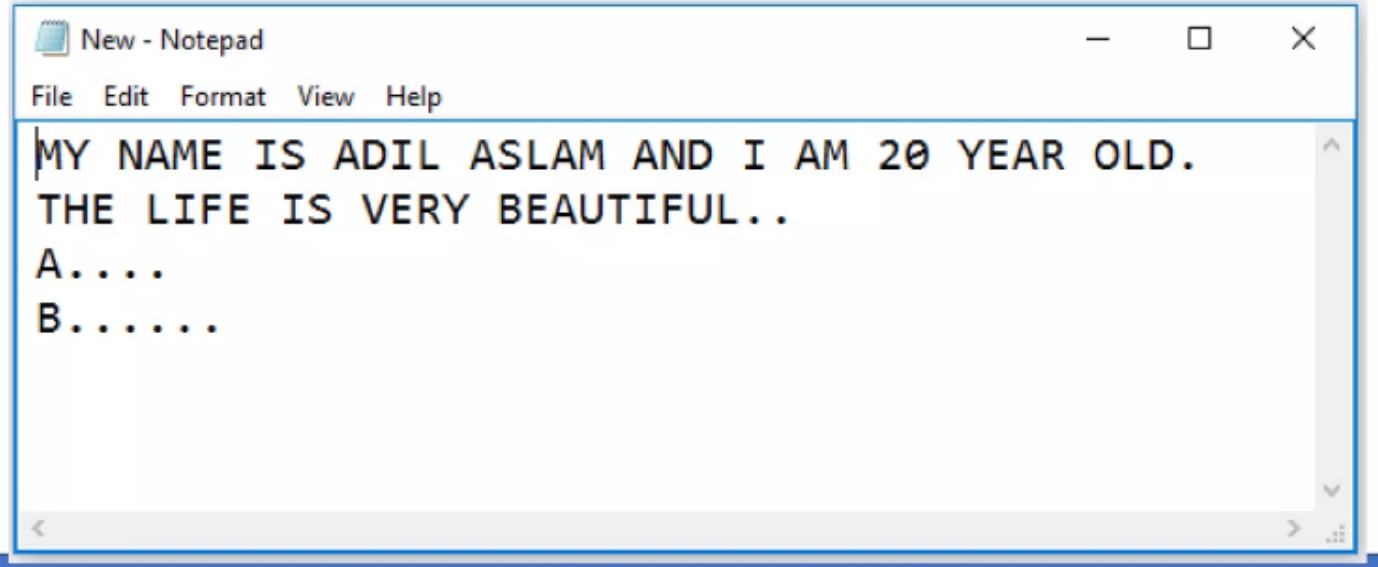
User Defined Function in C++ Named copyupper(), that reads the file String.TXT and Creates a new file named New.TXT contains all words from the file String.TXT in Uppercase

Solution of the Previous Problem

```
void copyupper() {
        ifstream fin;
        fin.open("String.TXT");
        ofstream fout;
        fout.open("New.TXT");
        char ch;
        while(!fin.eof()) {
                fin.get(ch);
                ch=toupper(ch);
                fout<<ch;
        }
        fin.close();
        fout.close();
}
int main() {
        copyupper();
}
```

Output of the Previous Program is :

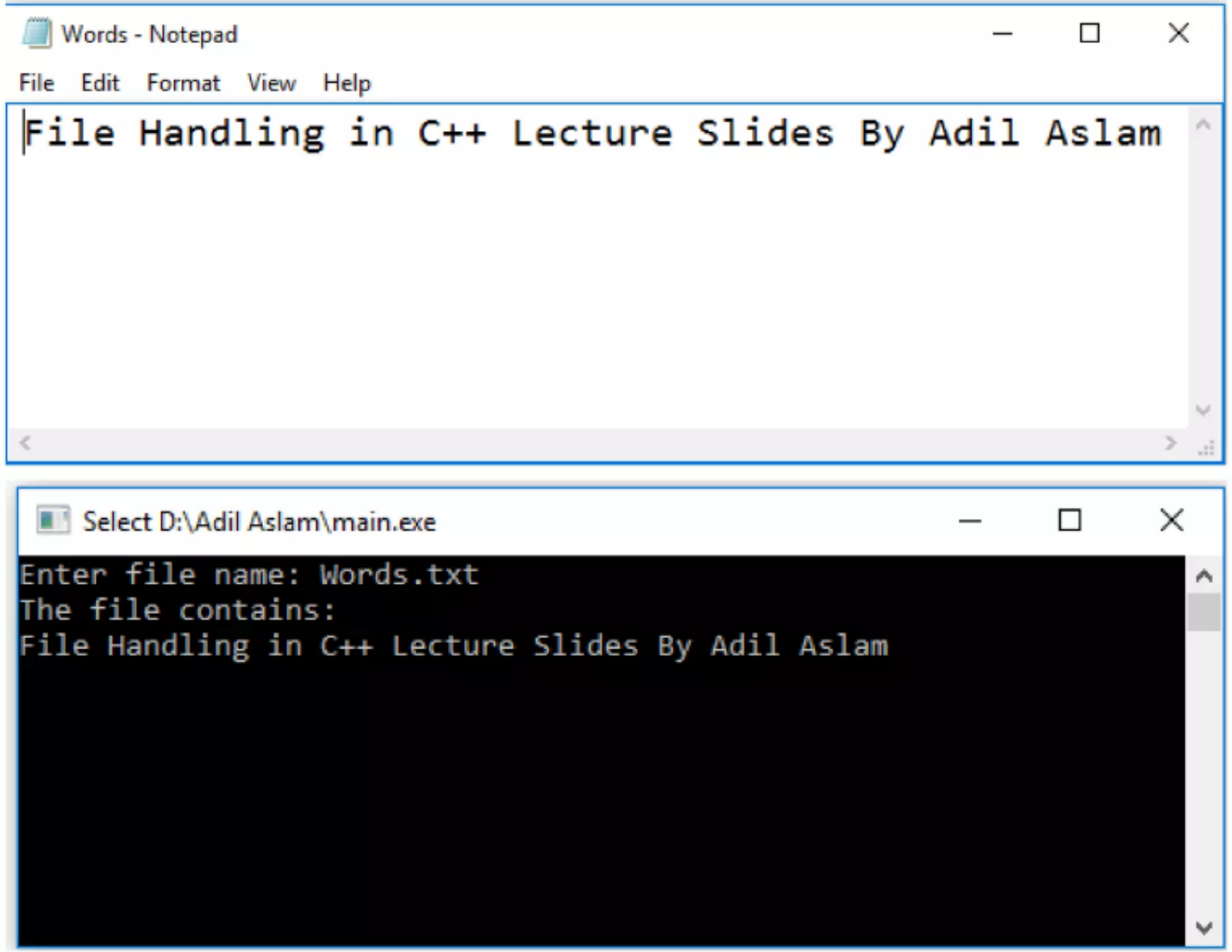




Retrieve Information from the file

```
int main() {
        ifstream fin;
        char fname[20];
        char rec[80];

        cout<<"Enter file name: ";
        cin>>fname;
        cout<<"The file contains:\n";
        fin.open(fname, ios::in);
        fin.get(rec, 80);
        cout<<rec;
        fin.close();
        getch();    //Require #include<conio.h>
}
```

Output of the Previous Program is :



Your Task.. ☺

- Write a program creates a file (entered by user) and store some content (entered by user). Then display those content (if user want) on the output screen in C++

Solution of the Previous Problem-1

```
#include<fstream>
#include<conio.h>
#include<iostream>
using namespace std;
int main() {
        char rec[80], ch;
        char fname[20];
        int count=0, i;
        char ans='y';
        cout<<"Enter file name: ";
        cin.get(fname, 20);
        ofstream fout(fname, ios::out);
```

Solution of the Previous Problem-2

```
if(!fout) {
        cout<<"Error in creating the file..!!\n";
        cout<<"Press any key to exit...\n";
        getch();
        exit(1);
}
cin.get(ch);
cout<<"Enter information to store..\n";
while(ans=='y' || ans=='Y') {
        cin.get(rec, 80);
        fout<<rec<<"\n";
        cout<<"Want to enter more ? (y/n).. ";
        cin>>ans;
        count++;
        cin.get(ch);
}
```

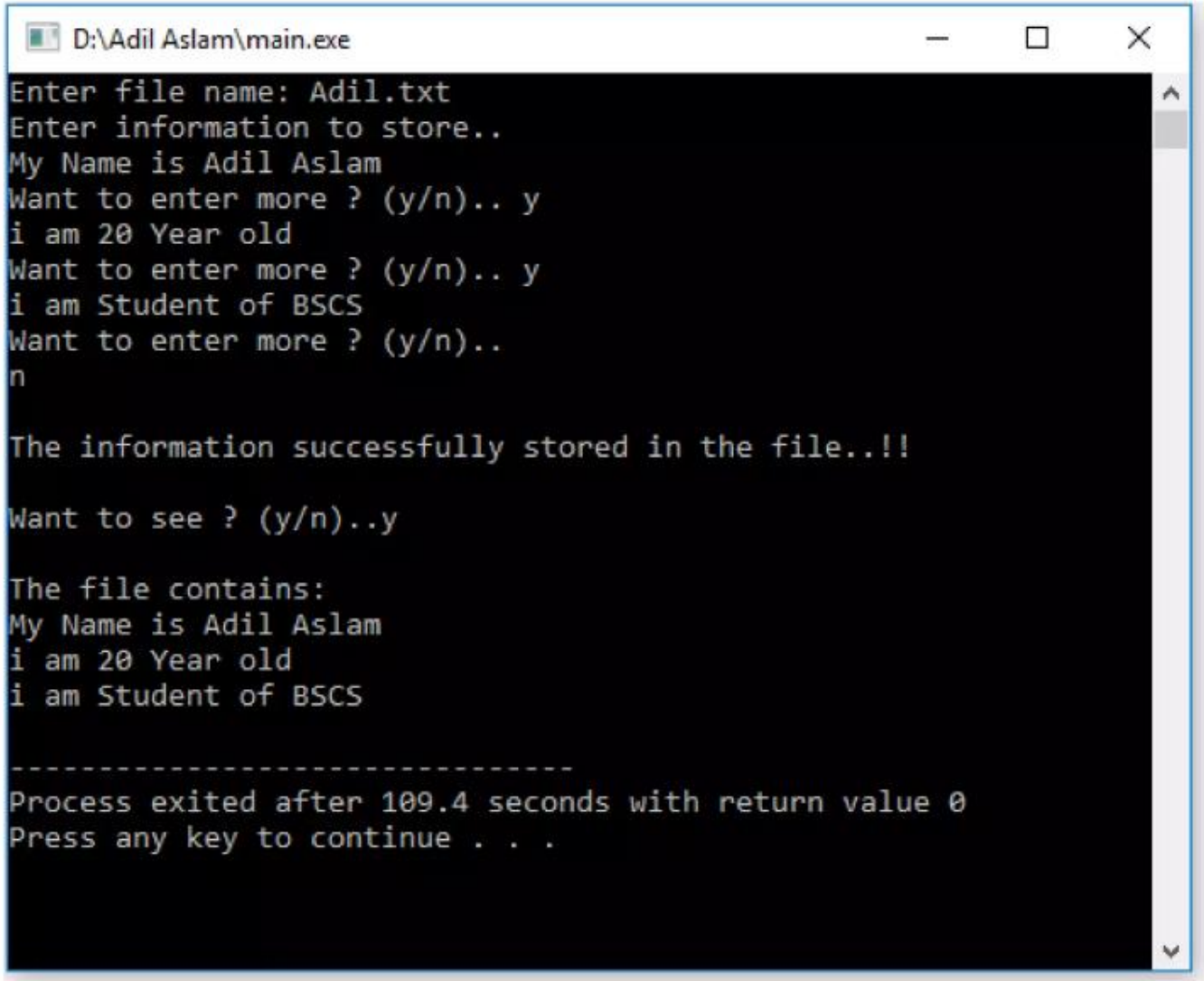
Solution of the Previous Problem-3

```
cout<<"\nThe information successfully stored in the file..!!\n";
        fout.close();
        cin.get(ch);
        cout<<"Want to see ? (y/n)..";
        cin>>ans;
        if(ans=='y' || ans=='Y')
        {
                ifstream fin(fname, ios::in);
                if(!fin) {
                        cout<<"Error in opening the file..!!\n";
                        cout<<"Press any key to exit..\n";
                        getch();
                        exit(2);
                }
```

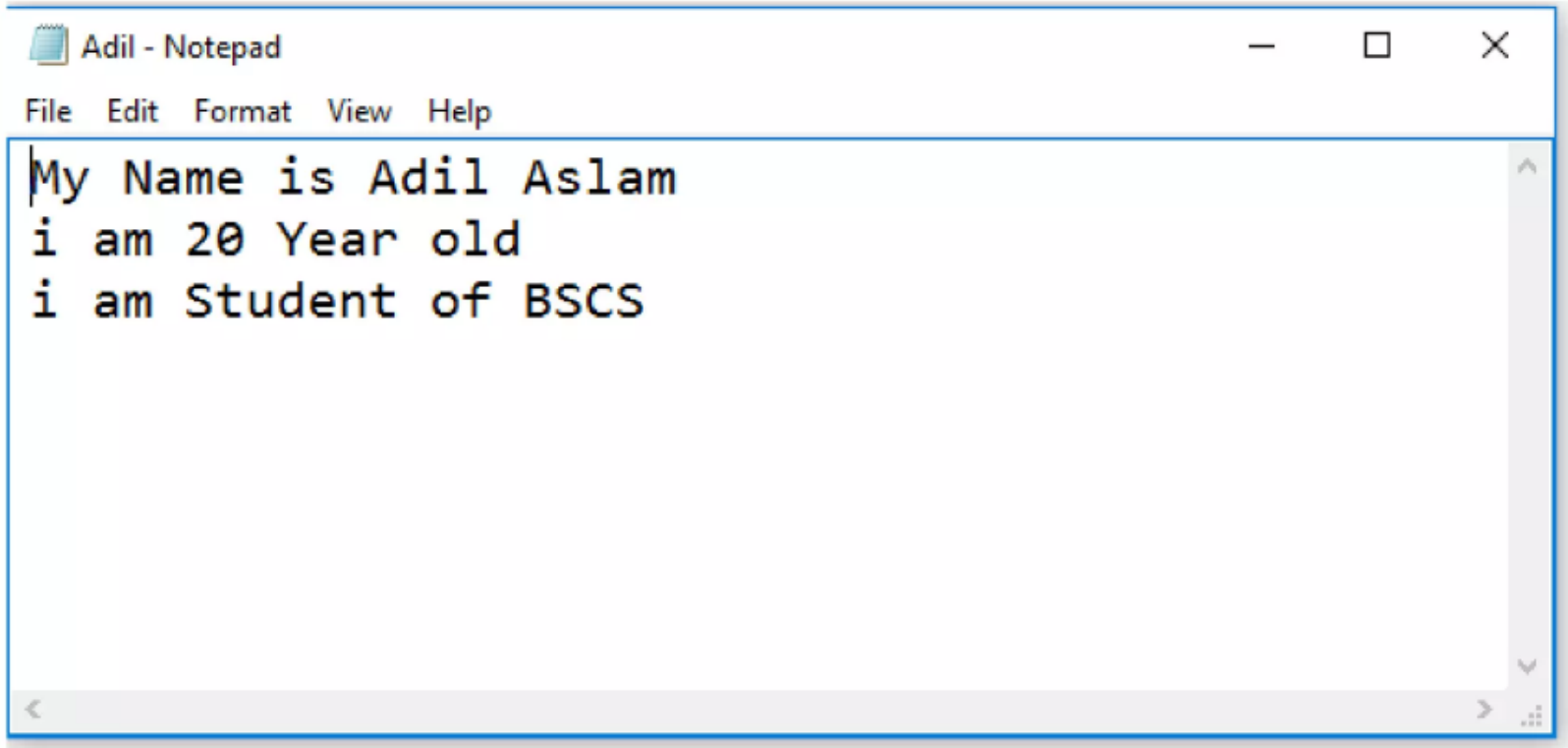
Solution of the Previous Problem-4

```
        //Moves the get pointer to a specific location in the file
        fin.seekg(0);
                cout<<"\n";
                cout<<"The file contains:\n";
                for(i=0; i<count; i++)
                {
                        fin.get(rec, 80);
                        fin.get(ch);
                        cout<<rec<<"\n";
                }
                fin.close();
         }

         getch();
}
```

Output of the Previous Program is :



Output of the Previous Program is :



Types of File Access

- ***Sequential access***. With this type of file access one must read the data in order, much like with a tape, whether the data is really stored on tape or not.

- ***Random access*** (or *direct access*). This type of file access lets you jump to any location in the file, then to any other, etc., all in a reasonable amount of time.

File Pointers

- Each file object has two integer values associated with it :
  - **get pointer**
  - **put pointer**
- These values specify the byte number in the file where reading or writing will take place.

File Pointers

- By default reading pointer is set at the beginning and writing pointer is set at the end (when you open file in ios::app mode)

- There are times when you must take control of the file pointers yourself so that you can read from and write to an arbitrary location in the file.

Random File Access

- ***Get* and *Put* Stream Pointers:**
  - Tellg()
  - Tellp()
  - Seekg()
  - Seekp()
- **Binary Files input and Output:**
  - Write
  - Read

Functions Associated with file Pointers

- The **seekg()** and **tellg()** functions allow you to set and examine the get pointer.
- The **seekp()** and **tellp()** functions allow you to set and examine the put pointer.

Get and *Put* Stream Pointers

- ***Get* Stream Pointers:**
  - Tellg()
  - Seekg()
- ***Put* Stream Pointers:**
  - Tellp()
  - Seekp()

Get and *Put* Stream Pointers

- ***Get* Stream Pointers:**
  - We used getstream pointer in ifstream.(input)
  - It gets the position of the element that to be read in file.
- ***Put* Stream Pointers:**
  - We used putstream pointer in ofstream.
  - It puts (or changes) the position of the element that to be written in file.

Function for Manipulation of file Pointer

Function	Operation
seekg()	moves get pointer (input) to a specified location.
seekp()	moves put pointer (output) to a specified location.
tellg()	gives the current position of the get pointer.
tellp()	gives the current position of the put pointer.
fout . seekg(0, ios :: beg)	go to start
fout . seekg(0, ios :: cur)	stay at current position
fout . seekg(0, ios :: end)	go to the end of file
fout . seekg(m, ios :: beg)	move to m+1 byte in the file
fout . seekg(m, ios :: cur)	go forward by m bytes from the current position
fout . seekg(-m, ios :: cur)	go backward by m bytes from the current position
fout . seekg(-m, ios :: end)	go backward by m bytes from the end

Two types of Member Functions of *Get* Stream

- ***Tellg()***
  - It tells the current location of the **Get** stream pointer.
  - It has no parameter but returns as **pos_type.**
- ***Seekg()***
  - It seeks to (or changes) the current position of the Get stream pointer.
  - *This function is overloaded and has perimeters and have return type of pos_type.*

Two types of Member Functions of *Put* Stream

- ***Tellp()***
  - It tells the current location of the **Put** stream pointer.
  - It has no parameter but returns as **pos_type.**
- ***Seekp()***
  - It seeks to (or changes) the current position of the **Put** stream pointer.
  - *functions are overloaded and both have perimeters and have return type of pos_type*

Get Stream Function

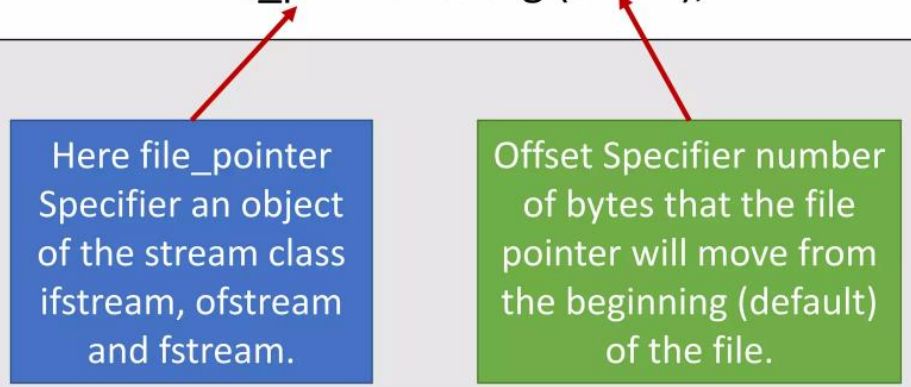
- **seekg()** is used to move the get pointer to a desired location.
- **Syntax:**

```
file_pointer.seekg (offset);
```

Get Stream Function

- **seekg()** is used to move the get pointer to a desired location.
- **Syntax:**

```
file_pointer.seekg (offset);
```



Example of seekg() Function

```
ifstream in;
in.open("Test.txt");
in.seekg(20);
```

In this example, the text file **Test** is opened in the read mode (default for the ifstream class).
In the **seekg()** function, the value 20 is passed as an argument, which specifies the offset value. Now, the input pointer moves to the byte number 20 in the file **Test** and the read operation starts the 21 byte of the file.

seekg() Function

- We can also use the seekg() function in the following way:

```
stream.seekg(offset, seek_direction);
```

Offset specifier the number of bytes that the file pointer has to move from the position, which is specified by the seek_direction

seekg() Function

- We can also use the seekg() function in the following way:

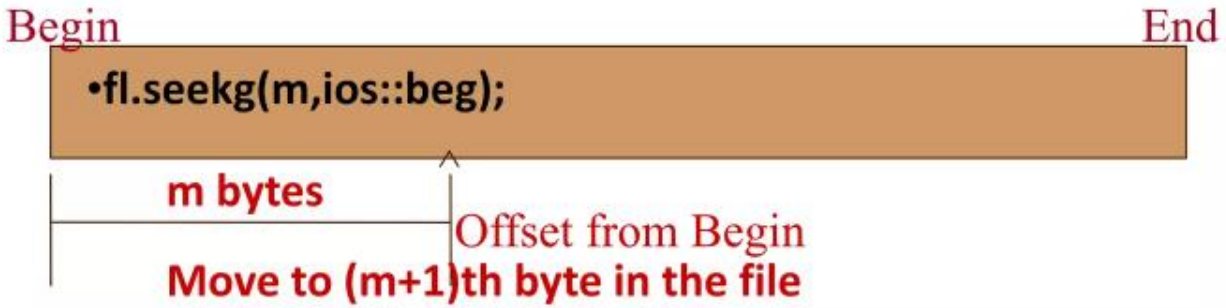
```
stream.seekg(offset, seek_direction);
```

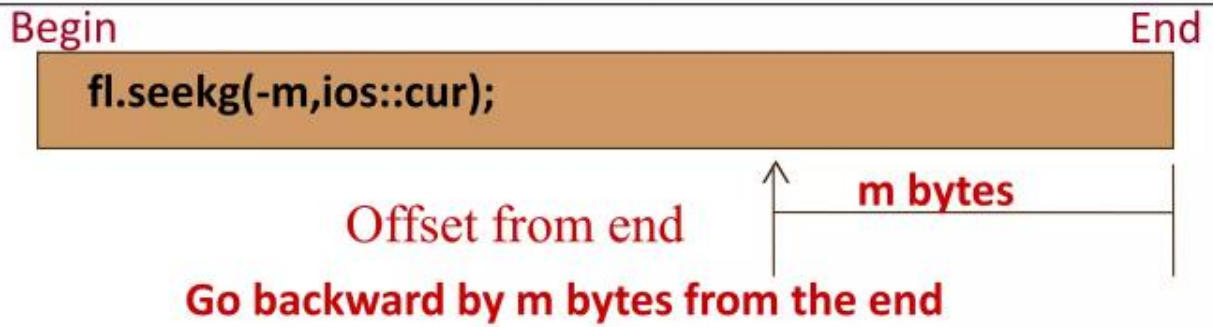
- The seek_direction take any value from the following:

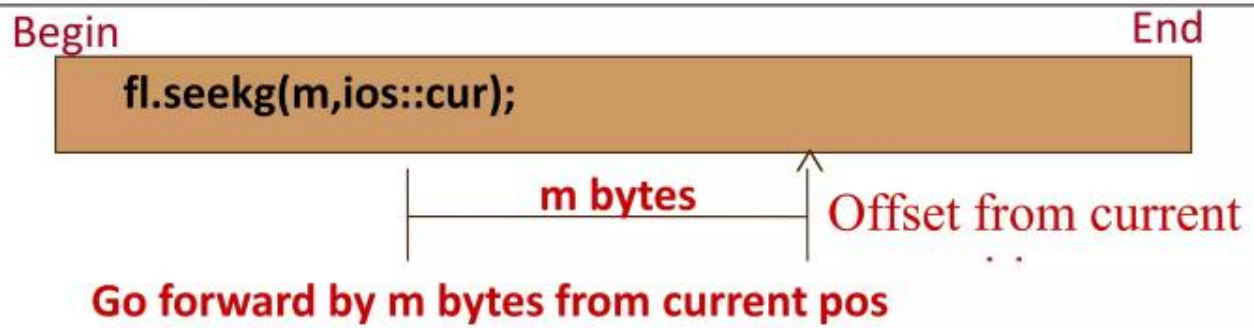
1	Ios::beg	//offset from the start of the file
2	Ios::cur	//offset from the current position of the pointer
3	Ios::end	//offset from the end of the file

The above three constants are defined in the ios class

seekg() function : (With two arguments)







seekg() Function Example

```
ifstream in;
in.open("Test");
in.seekg(0, ios::beg);
```

In the above example the ios::beg will set the output or the put pointer at the start of the file from where the reading operation begins.

seekg() Function Example

Now if we pass a negative value for the first argument(offset), the read or write operation will be in the backward direction from the position specified by the see_direction.

For Example:

```
ifstream in;
in.open("Test");
in.seekg(-2, ios::cur);
```

In the above example, the ios::cur sets the input pointer at the current position in the file.
Now it will move 2 bytes backward from the current position because the offset value is -2

The tellg() Function

- The tellg() function is used to get current position of the input pointer in the file.
- This function return an integer value.
- tellg() is used to know where the get pointer is in a file.
- **Syntax:**

```
file_pointer.tellg();
```

- **Example:**

```
int posn = fin.tellg();
```

The tellg() Function Example

```
ifstream in;
in.open("Test.txt");
in.seekg(10, ios::beg);
int num = in.tellg();
```

In the above example **seekg()** function moves the input pointer to the 10th byte from the start of the file Test.
The function **tellg()** return the current position of the input pointer and the resultant value is assigned to the variable **num**.

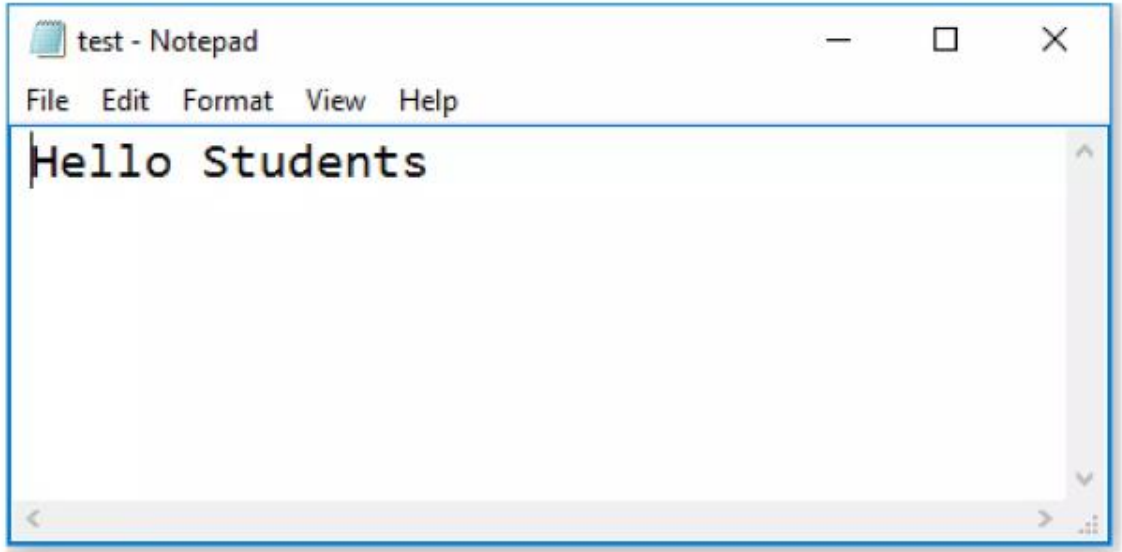
Program Example of seekg() and tellg()-1

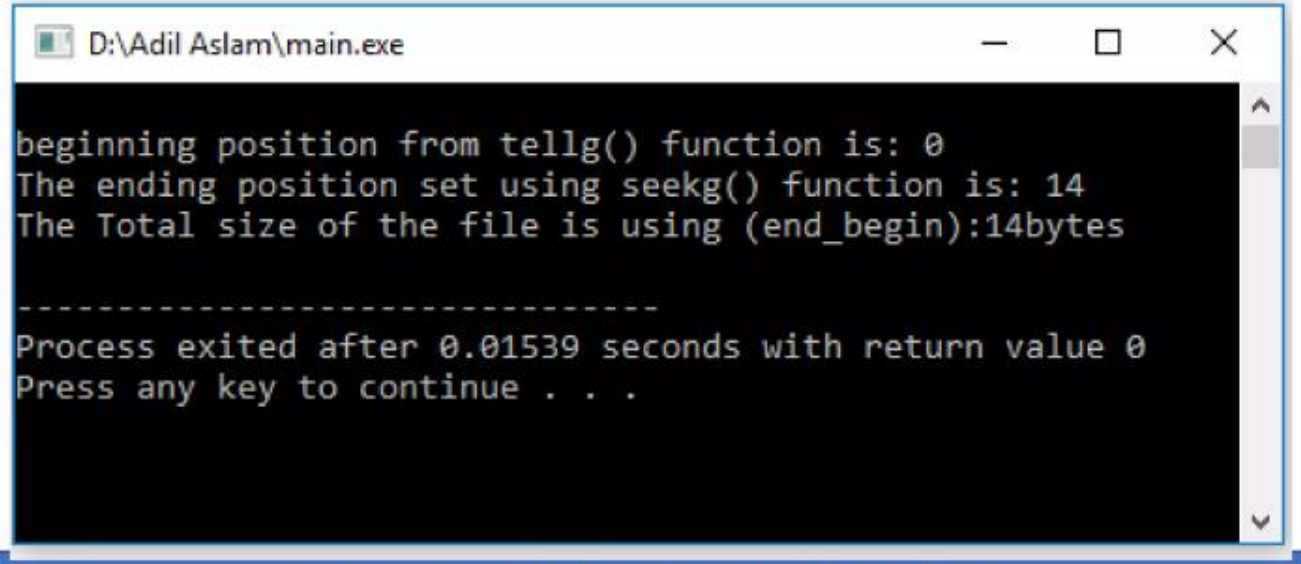
```
#include <iostream>
#include<fstream>
using namespace std;
int main()
{
    long begin, end;
    ifstream myfile("test.txt");
    //telling the current pointer of get stream to begin var.
    begin=myfile.tellg();
   //output result
  cout<<"\nbeginning position from tellg() function is:"<<begin<<endl;
```

Program Example of seekg() and tellg()-2

```
//seeking (changing) the current pointer of get stream to end var.
        myfile.seekg(0, ios::end);    //seekg (position. direction)
        end=myfile.tellg();
        //output result
        cout<<"The ending position set using seekg() function is:
        "<<end<<endl;
        myfile.close();   //closing file
        //output the size of the file
        cout<<"The Total size of the file is using (end_begin):"
        <<(end-begin)<<"bytes"<<endl;
        return 0;
}
```

Output of the Previous Program is :





The seekp() Function

- As we already know that the seekg() function is used to move the input pointer to a specified location.
- Similarly, the seekp() function is used to move the output pointer to a specified location.
- We can use the seekp() function in the same way as the seekg() function.

The seekp() Function Example

- **seekp()** is used to move the put pointer to a desired location with respect to a reference point.

- **Syntax:**

```
file_pointer.seekp(number of bytes  , Reference point);
```

- **Example**

```
fout.seekp(10 , ios::beg);
```

The seekp() Function Example

```
ofstream out;
out.open("Test");
out.seekp(0, ios::end)
```

In the above example the ios::end sets the output pointer at the end of the file

The tellp() Function

- The tellp() is used to get current position of the output pointer in the file.
- This function also return an integer value.
- **tellp()** is used to know where the put pointer is in a file.
- **Syntax:**

```
file_pointer.tellp();
```

- **Example:**

```
int posn=fout.tellp();
```

The tellp() Function Example

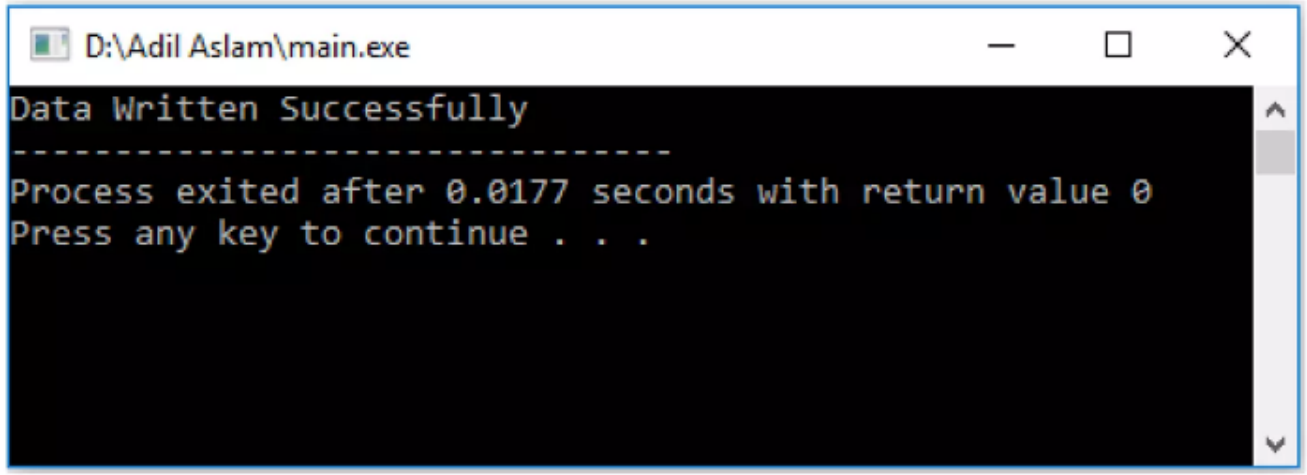
```
ofstream out;
out.open("Test");
in.seekp(10, ios::beg);
int num = in.tellp();
```

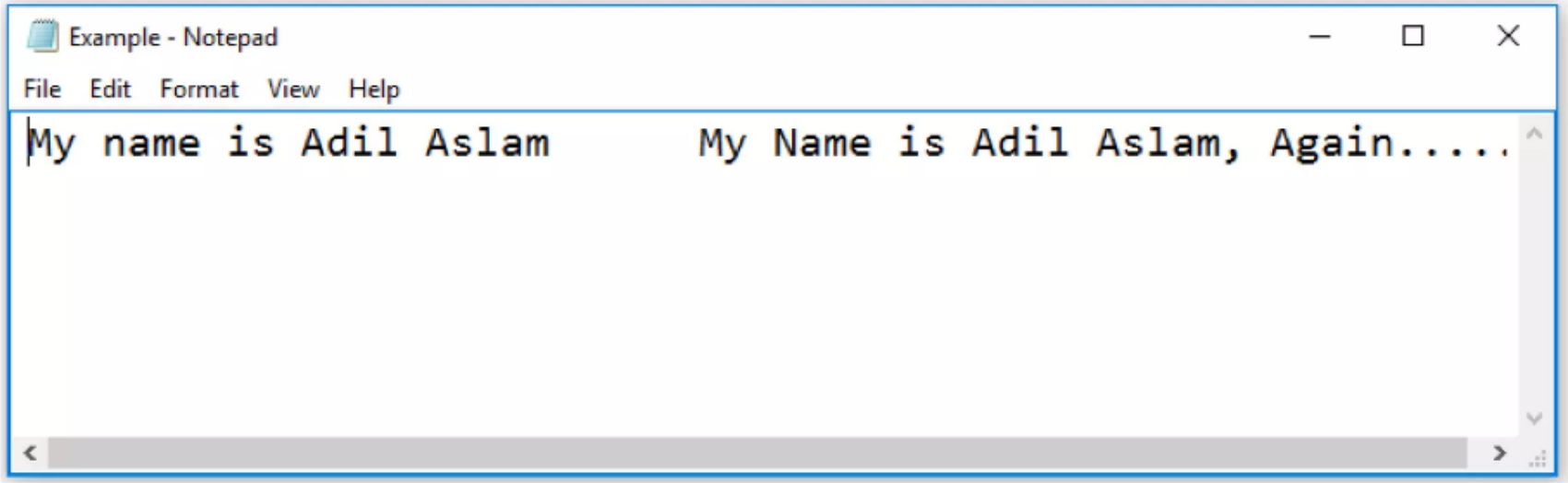
In the above example the **seep()** function move the output pointer to the 10^{th} byte from the start of the file Test.
The function **tellp()** return the current position of the output pointer and the resultant value is assign to the variable num.

Program Example of seekp() and tellp()

```
int main() {
        ofstream outfile;
        //open file
        outfile.open("Example.txt");
        //writing some line to file
        outfile<<"My name is Adil Aslam";
         //tells the current position of the pointer
        long pos = outfile.tellp();
        //change the current position of the pointer
        outfile.seekp(pos+6);
        //writing something to file
        outfile<<"My Name is Adil Aslam, Again.....";
        outfile.close();
        cout<<"Data Written Successfully";
        return 0;
}
```

Output of the Previous Program is :





Mix Example of both Get and put Stream Functions-1

```
#include<iostream>
#include<fstream>
using namespace std;
int main() {
  fstream fp;
  char buf[100];
  int pos;
  // open a file in write mode with 'ate' flag
  fp.open("random.txt", ios :: out | ios :: ate);
  cout << "\nWriting to a file ... " << endl;
```

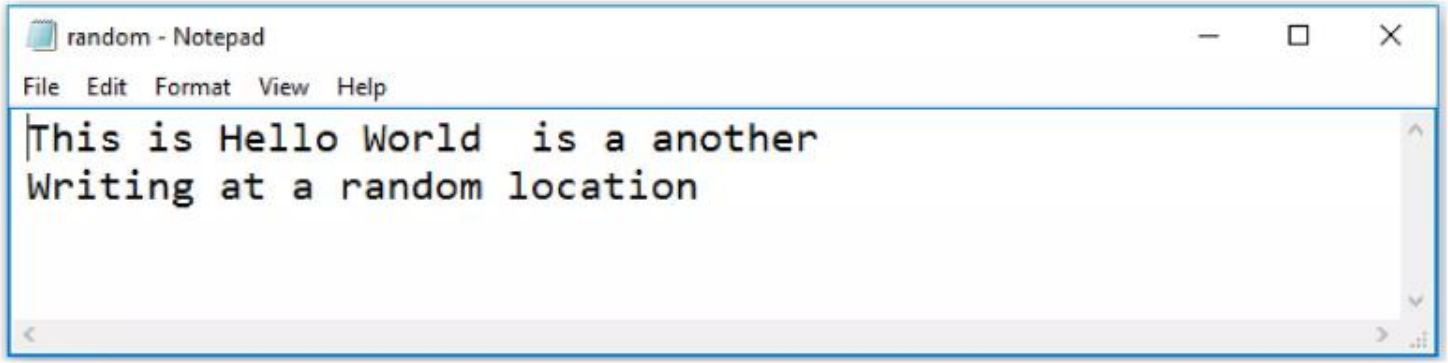
Mix Example of both Get and put Stream Functions-2

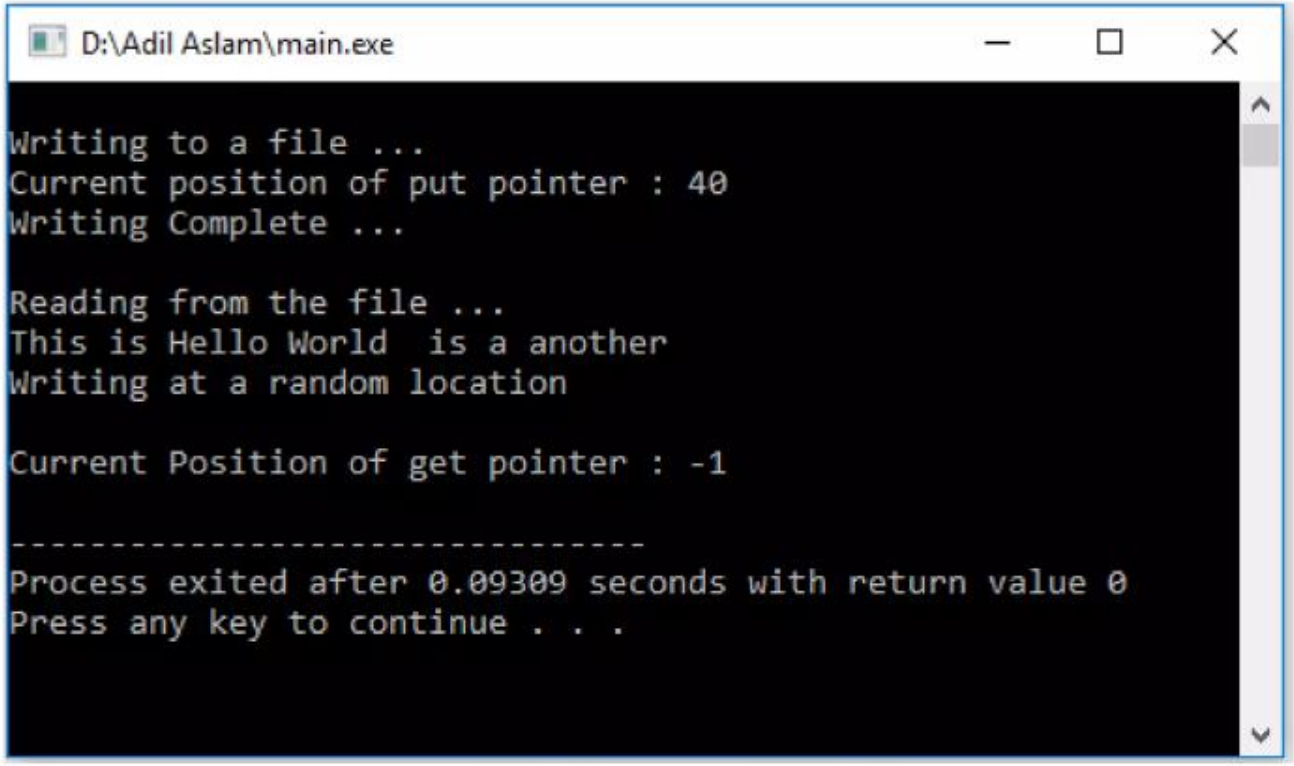
```
fp << "This is a line" << endl;     // write a line to a file
 fp << "This is a another line" << endl;   // write another file
 pos = fp.tellp();
 cout << "Current position of put pointer : " << pos << endl;
 // move the pointer 7 bytes backward from current position
 fp.seekp(-7, ios :: cur);
 fp << endl << "Writing at a random location  ";
 // move the pointer 7 bytes forward from beginning of the file
 fp.seekp(7, ios :: beg);
 fp << " Hello World  ";
 fp.close();    // file write complete
 cout << "Writing Complete ... " << endl;
```

Mix Example of both Get and put Stream Functions-3

```
// open a file in read mode with 'ate' flag
  fp.open("random.txt", ios :: in | ios :: ate);
  cout << "\nReading from the file ... " << endl;
  fp.seekg(0); // move the get pointer to the beginning of the file
  // read all contents till the end of file
  while (!fp.eof()) {
     fp.getline(buf, 100);
     cout << buf << endl;
  }
  pos = fp.tellg();
  cout << "\nCurrent Position of get pointer : " << pos << endl;
  return 0; }
}
```

Output of the Previous Program is :





Calculate Average of Numbers-1

```
#include <fstream>
#include <iostream>
#include <cstring>
using namespace std;
int main()
 {
        ofstream fout;
        //ifstream fin("Values.txt"); // declare an input file stream
        ifstream fin; // declare an input file stream
        string name;
        int x = 0;
        double avg = 0.0;
        int sum = 0;
        cout << "Enter file name: ";
        cin >> name;
```

Calculate Average of Numbers-2

```
// open file file_name for input
fin.open(name.c_str(), ios::in);
// check if file is opened for input
if (!fin.is_open())
{
        cerr << "Unable to open file.." << endl;
        exit(10);
}
// read text from file
int ne = 0;
int oe = 0;
```

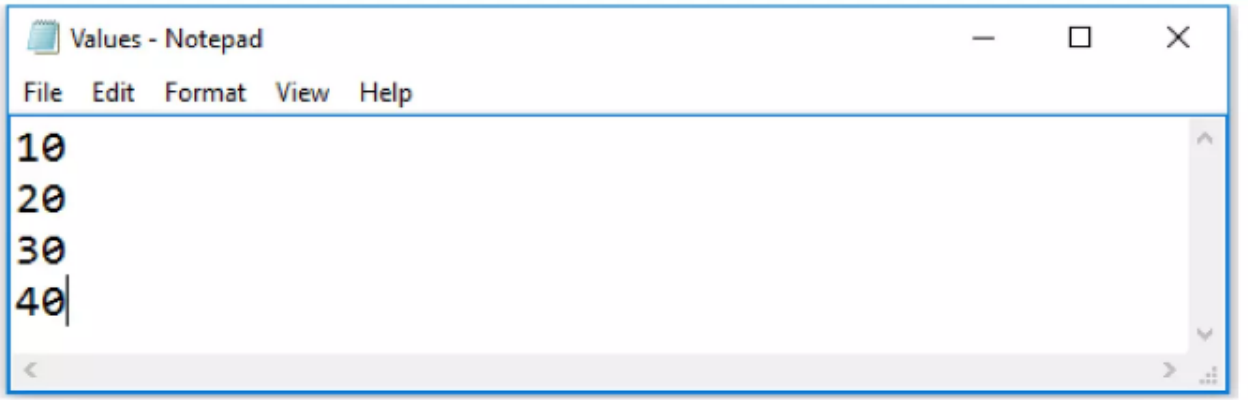
Calculate Average of Numbers-3

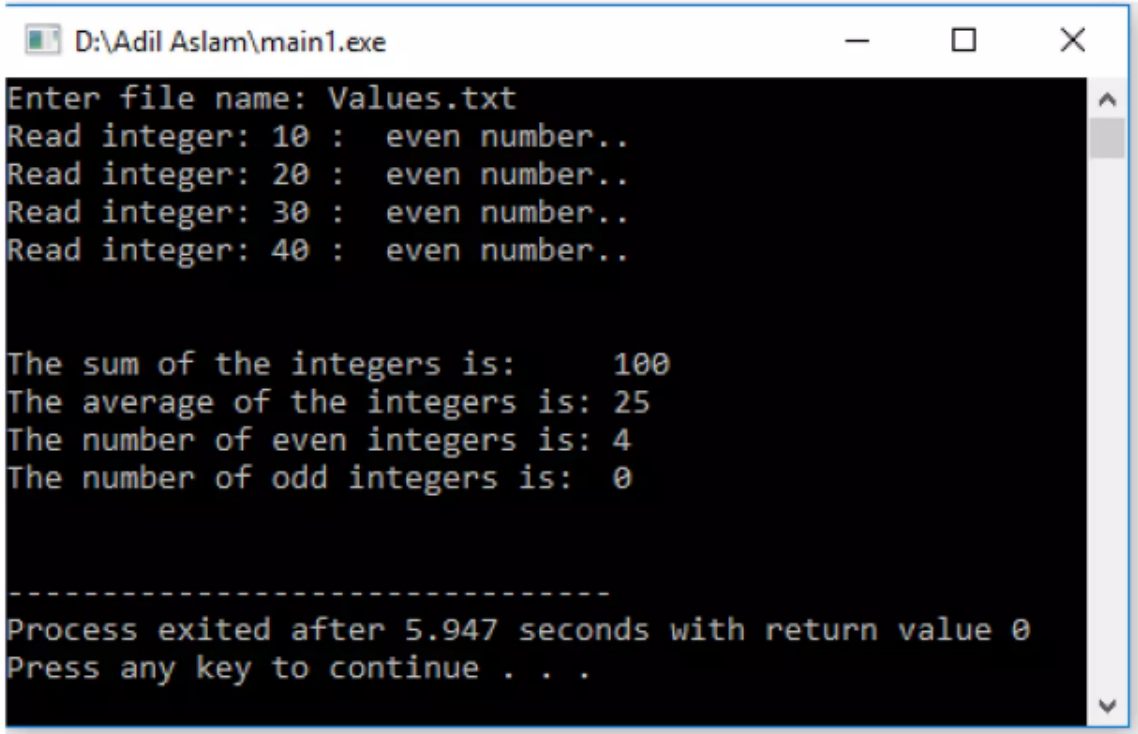
```
while (!fin.fail()) {
        fin >> x;
        if (fin) {
                cout << "Read integer: " << x << " : ";
                sum += x;
                if (x%2 == 0) {
                        cout << " even number.." << endl;
                        ne++;
                }
                else {
                        cout << " odd number.." << endl;
                        oe++;
                }
        } //end outer if
} //end of while loop
```

Calculate Average of Numbers-4

```
        avg = sum/(ne+oe);
        cout << endl << endl;
        cout << "The sum of the integers is:     " << sum << endl;
        cout << "The average of the integers is: " << avg << endl;
        cout << "The number of even integers is: " << ne << endl;
        cout << "The number of odd integers is:  " << oe << endl << endl;
        if (!fin.eof()) { // check for error
                cerr << "Error reading file." <<  endl;
                exit(20);
        }
        fin.close();

        return 0;
}
```

Output of the Previous Program is :





Searching in File-1

```
#include<iostream>
#include<fstream>
#include<conio.h>
using namespace std;
class student
{
        int rollno;
        char name[20];
        char branch[3];
        float marks;
        char grade;
```

Searching in File-2

```
public:
        void getdata()
        {
                cout<<"Rollno: ";
                cin>>rollno;
                cout<<"Name: ";
                cin>>name;
                cout<<"Branch: ";
                cin>>branch;
                cout<<"Marks: ";
                cin>>marks;
```

Searching in File-3

```
            if(marks>=75){
                    grade = 'A';
            }
            else if(marks>=60) {
                    grade = 'B';
            }
            else if(marks>=50) {
                    grade = 'C';
            }
            else if(marks>=40) {
                    grade = 'D';
            }
            else{
                    grade = 'F';
            }
    }
```

Searching in File-4

```
void putdata()
                {
                cout<<name<<", rollno "<<rollno<<" has ";
                cout<<marks<<"% marks and "<<grade
                <<" grade."<<"\n";
                }

                int getrno()
                {
                        return rollno;
                }
}stud1;
```

Searching in File-5

```
int main() {
        ofstream fout("marks.dat", ios::out);
        char ans='y';
        while(ans=='y' || ans=='Y')
        {
                stud1.getdata();
                fout.write((char *)&stud1, sizeof(stud1));
                cout<<"Record added to the file\n";
                cout<<"\nWant to enter more ? (y/n)..";
                cin>>ans;
        }
        fout.close();
         int rno;
        char found;
```

Searching in File-6

```
ifstream fin("marks.dat", ios::in);
found = 'n';
cout<<"Enter rollno to be searched for: ";
cin>>rno;
while(!fin.eof())  {   // end-of-file used here
        fin.read((char *)&stud1, sizeof(stud1));
        if(stud1.getrno() == rno) {
        cout<<"Record found at roll number "<<rno
         <<". Here is the record\n";
                stud1.putdata();
                found = 't';
                break;
        }
}
```

Searching in File-7

```
        if(found=='n')
        {
                cout<<"\nRecord not found at this roll
                number..!!\n";
                cout<<"Press any key to exit...\n";
                getch();
                exit(2);
        }

        fin.close();
        cout<<"\nPress any key to exit...\n";
        getch();
}
```

Output of the Previous Program is :