Lecture 20

Filing & Structs

UBS
ITU INFORMATION TECHNOLOGY UNIVERSITY

QUIZ

سورة طه پارہ: 16

قَالَ رَبِّ اشْرَحْ لِيْ صَدْرِيْ ۙ ﴿٢٥﴾

[قَالَ: اس نے کہا] [رَبِّ: اے رب !] [اشْرَحْ: کھول دے] [لِيْ: میرے لیے] [صَدْرِيْ : میرا سینہ]

وَيَسِّرْ لِيْٓ اَمْرِيْ ۙ ﴿٢٦﴾

[وَيَسِّرْ: اور آسان کر دے] [لِيْ: میرے لیے] [اَمْرِيْ : میرا کام]

وَاحْلُلْ عُقْدَةً مِّنْ لِّسَانِيْ ۙ ﴿٢٧﴾

[وَاحْلُلْ: اور کھول دے] [عُقْدَةً: گرہ] [مِّنْ: سے] [لِّسَانِيْ: میری زبان]

يَفْقَهُوْا قَوْلِيْ ۖ ﴿٢٨﴾

[يَفْقَهُوْا: وہ سمجھ سکیں] [قَوْلِيْ : میری بات]

QUESTIONS / FEEDBACK / CONCERNS

UBS

ITU INFORMATION TECHNOLOGY UNIVERSITY

SE SEC A
SLIDE OF FAME

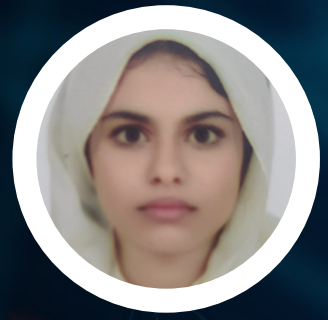
NO ONE
WEEK-1

Muhammad Daniyal Hammad (BSSE23046)
WEEK-2

Syed Hashim Abbas (BSSE23084)
WEEK-3

Umar Ahmad (BSSE23032)
WEEK-4

Umar Ahmad (BSSE23032)
WEEK-5



Fatima Noorulain BSSE23003
WEEK-6

Umar Ahmad (BSSE23032)
WEEK-7

YOUR NAME
WEEK-8

YOUR NAME
WEEK-9

YOUR NAME
WEEK-10

YOUR NAME
WEEK-11

YOUR NAME
WEEK-12

YOUR NAME
WEEK-13

YOUR NAME
WEEK-14

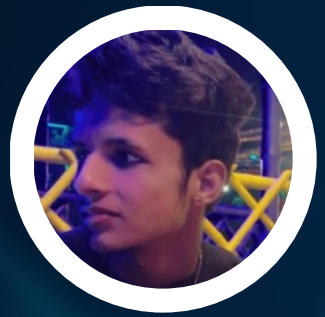
YOUR NAME
WEEK-15

UBS

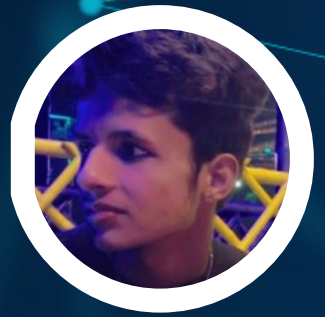
ITU INFORMATION TECHNOLOGY UNIVERSITY

SE SEC B
SLIDE OF FAME

Muhammad Mukarram
BSSE23029
WEEK-1



Muhammad Abdullah
(BSSE23087)
WEEK-2



Muhammad Abdullah
(BSSE23087)
WEEK-3

Fasiha Rohail
(BSSE23041)
WEEK-4

Muhammad Abdullah
(BSSE23087)
WEEK-5

Hazira Azam
BSSE23019
WEEK-6

Jamshaid Ahmed
BSSE23012
WEEK-7

YOUR NAME
WEEK-8

YOUR NAME
WEEK-9

YOUR NAME
WEEK-10

YOUR NAME
WEEK-11

YOUR NAME
WEEK-12

YOUR NAME
WEEK-13

YOUR NAME
WEEK-14

YOUR NAME
WEEK-15

UBS

ITU INFORMATION TECHNOLOGY UNIVERSITY

RECAP

GitHub

Tools (Cygwin, IDE, GitHub)

Approach towards a word problem

Flowcharts

Flowcharts Advantages & Disadvantages

Algorithms

Pseudocode

Numbers Systems (Decimal, Binary, Octal & Hexadecimal)

Ten’s Complement

Twos Complement

main function

Stream in and stream out operators

if else

Functions

Data Types

Arithmetic Operators

Relational Operators

Loops (While, for , do while)

Nested Loops

Switch cases

RECAP

Function Overloading

Scope of variables

Function Prototype and Definition

Default Value in parameters of functions

Parameters by value vs Parameters by Reference

Recursion

Arrays

2D Arrays / Multi Dimensional Arrays

Pointers

File

A file is a collection of related data stored in a particular area on the disk. The data is stored in disk using the concept of file .

Why File

Permanent storage of data : - (all the message or value printed with help of any output statements like printf , putchar are never available for future use) .

If there is a large amount of data generated as output by a program, storing that output in file will help in easy handling /analysis of the output , as user can see the whole output at any time even after complete execution of the program.

If we need lot of data to be inputted, user cannot keep on typing that again and again for repeated execution of program. In that case, all input data can be once written in a file and then that file can be easily used as the input file.

The transfer of input – data or output – data from one computer to another can be easily done by using files.

Using Input/Output Files

A computer file

- is stored on a secondary storage device (e.g., disk);
- is permanent;
- can be used to
  - provide input data to a program
  - or receive output data from a program
  - or both;
- should reside in Project directory for easy access;
- must be opened before it is used.

General File I/O Steps

1. Include the header file fstream in the program.
2. Declare file stream variables.
3. Associate the file stream variables with the input/output sources.
4. Open the file
5. Use the file stream variables with >>, <<, or other input/output functions.
6. Close the file.

Using Input/Output Files

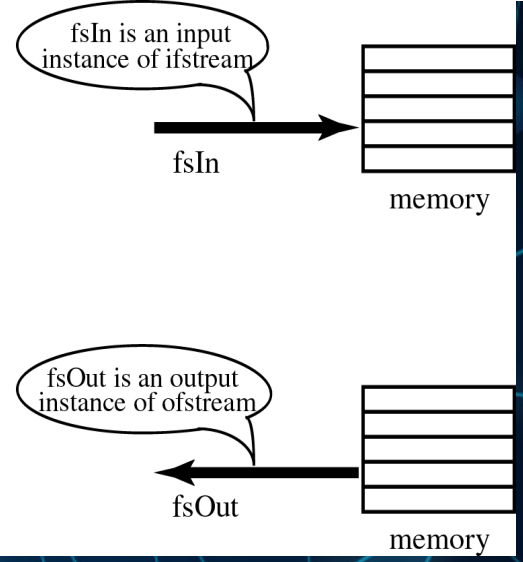
- ***stream*** - a sequence of characters
  - interactive (iostream)
    - **cin** - input stream associated with **keyboard.**
    - **cout** - output stream associated with **display**
  - file (fstream)
    - **ifstream** - defines new input stream (normally associated with a file).
    - **ofstream** - defines new output stream (normally associated with a file).

Stream I/O Library Header Files

- Note: There is no ".h" on standard header files : <fstream>
- iostream -- contains basic information required for all stream I/O operations
- fstream -- contains information for performing file I/O operations

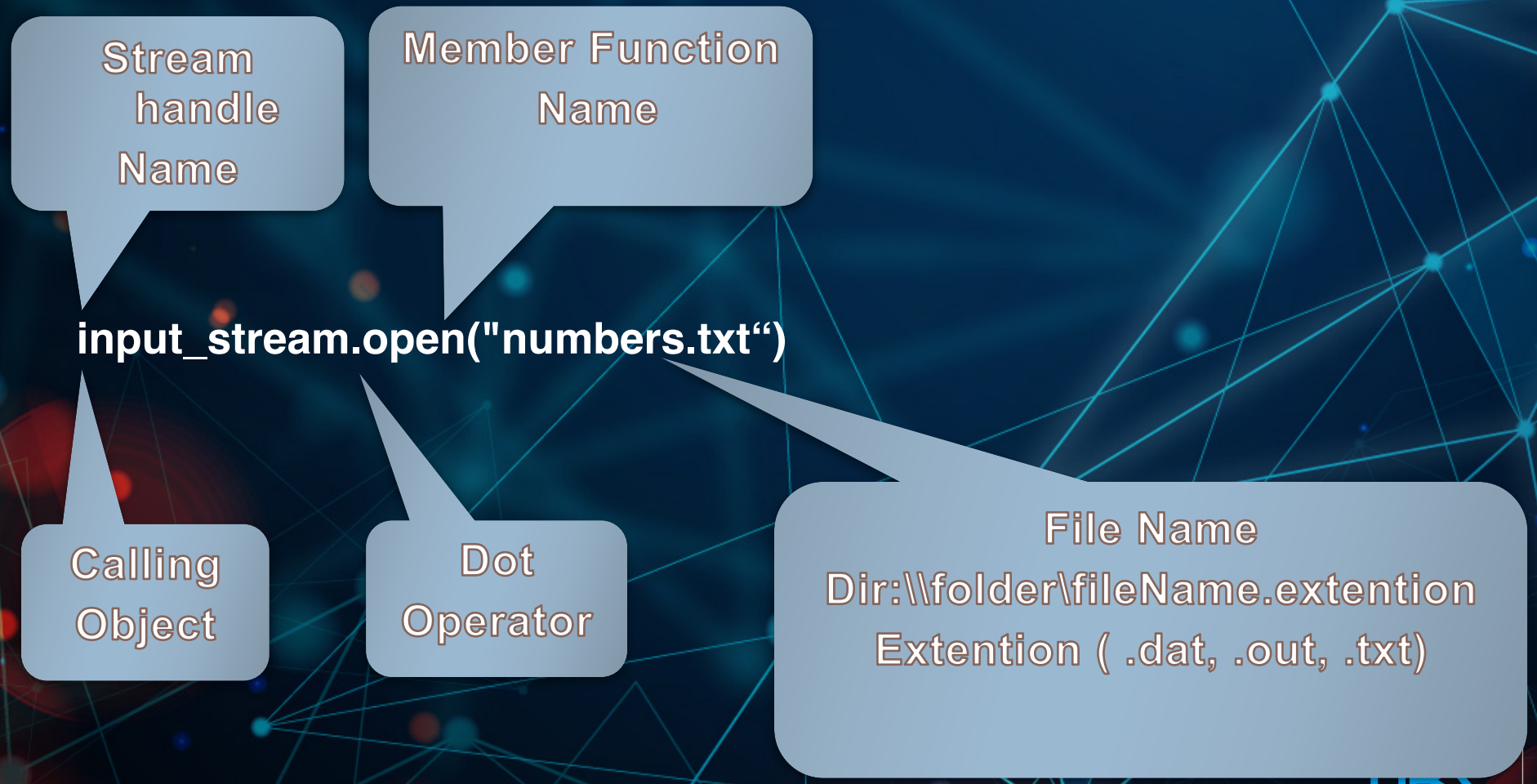
```
//Add additional header files you use
#include <fstream>
int main ()
{ /* Declare file stream variables such as
 the following  */
    ifstream  fsIn;//input
    ofstream fsOut; // output
    fstream both //input & output
//Open the files
    fsIn.open("prog1.txt"); //open the input file
    fsOut.open("prog2.txt"); //open the output file
//Code for data manipulation
.
.
//Close files
    fsIn.close();
    fsOut.close();
    return 0;
}
```

```
#include <fstream.h>

int main (void)
{
// Local Declarations
   ifstream      fsIn;

   ofstream      fsOut;
   •
   •
   •

}  // main
```



INFORMATION TECHNOLOGY UNIVERSITY

Object and Member Functions



Open()

- Opening a file associates a file stream variable declared in the program with a physical file at the source, such as a disk.
- In the case of an input file:
  - the file must exist before the open statement executes.
  - If the file does not exist, the open statement fails and the input stream enters the fail state
- An output file does not have to exist before it is opened;
  - if the output file does not exist, the computer prepares an empty file for output.
  - If the designated output file already exists, by default, the old contents are erased when the file is opened.

Validate the file before trying to access

By checking the stream variable;

```
If ( ! Mystream)
{
Cout << “Cannot open file.\n ”;
}
```

By using bool is_open() function.

```
If ( ! Mystream.is_open()) {
Cout << “File is not open.\n ”;
}
```

File I/O Example: Open the file with validation

```
#include <fstream>
using namespace std;
int main()
{
//declare output file variable
    ofstream outFile;
// open an exist file fout.txt
    outFile.open("fout.txt");
// Open validation
    if(! outFile.is_open() ) {
            Cout << "Cannot open file.\n ";
            return 1;
    }
    return 0;
}
```

File I/O Example: Open the file with validation

```
#include <fstream>
using namespace std;
int main()
{
//declare and automatically open the file
    ofstream outFile(“fout.txt");
// Open validation
    if(! outFile) {
        Cout << “Cannot open file.\n ”;
        return 1;
    }
    return 0;
}
```

More Input File-Related Functions

- **ifstream fsin;**
- **fsin.open(const char[] fname)**
  - connects stream **fsin** to the external file ***fname.***
- **fsin.get(char character)**
  - extracts next character from the input stream **fsin** and places it in the character variable ***character.***
- **fsin.eof()**
  - tests for the end-of-file condition.

File I/O Example: Reading

```
#include <iostream>
#include <fstream>

int main()
{//Declare and open a text file
    ifstream openFile("data.txt");
    char ch;
 //do until the end of file
    while( ! openFile.eof() )
    {
        openFile.get(ch); // get one character
        cout << ch;   // display the character
    }
    openFile.close(); // close the file

    return 0;
}
```

File I/O Example: Reading

```
#include <iostream>
#include <fstream>
#include <string>
int main()
{//Declare and open a text file
      ifstream openFile("data.txt");
      string line;
      while(!openFile.eof())
      {//fetch line from data.txt and put it in a string
            getline(openFile, line);
            cout << line;
      }
      openFile.close(); // close the file
      return 0;
}
```

More Output File-Related Functions

- **ofstream fsOut;**
- **fsOut.open(const char[] fname)**
  - connects stream **fsOut** to the external file ***fname.***
- **fsOut.put(char character)**
  - inserts character ***character*** to the output stream **fsOut.**
- **fsOut.eof()**
  - tests for the end-of-file condition.

File I/O Example: Writing

```
#include <fstream>
using namespace std;
int main()
{// declare output file variable
     ofstream outFile;
// open an exist file fout.txt
    outFile.open("fout.txt”);
//behave just like cout, put the word into  the
file
      outFile << "Hello World!";
      outFile.close();
       return 0;
}
```

File I/O Example: Writing

```
#include <fstream>
using namespace std;
int main()
{/* declare and automatically open the
file*/

     ofstream outFile("fout.txt");
//behave just like cout, put the word into
the file

     outFile << "Hello World!";
     outFile.close();


return 0;
}
```

File Open Mode

Name	Description
ios::in	Open file to read
ios::out	Open file to write
ios::app	All the data you write, is put at the end of the file. It calls ios::out
ios::ate	All the data you write, is put at the end of the file. It does not call ios::out
ios::trunc	Deletes all previous content in the file. (empties the file)
ios::nocreate	If the file does not exists, opening it with the open() function gets impossible.
ios::noreplace	If the file exists, trying to open it with the open() function, returns an error.
ios::binary	Opens the file in binary mode.

File Open Mode

```
#include <fstream>
int main(void)
{
ofstream outFile("file1.txt", ios::out);
outFile << "That's new!\n";
outFile.close();
        Return 0;
}
```

If you want to set more than one open mode, just use the **OR** operator- **|**. This way:

ios::ate | ios::binary

```
  11101101
| 00010010
= 11111111
```

```
  00001101
| 00010011
= 00011111
```

```
  100000000
| 01000000
= 110000000
```

Summary of Input File-Related Functions

```
#include <fstream>
ifstream fsIn;
```

- **`fsIn.open(const char[] fname)`**
  - connects stream **`fsIn`** to the external file ***`fname.`***
- **`fsIn.get(char& c)`**
  - extracts next character from the input stream **`fsIn`** and places it in the character variable ***`c.`***
- **`fsIn.eof()`**
  - tests for the end-of-file condition.
- **`fsIn.close()`**
  - disconnects the stream and associated file.
- **`fsIn >> c;`** //Behaves just like cin

Summary of Output File-Related Functions

```
#include <fstream>
ofstream fsOut;
```

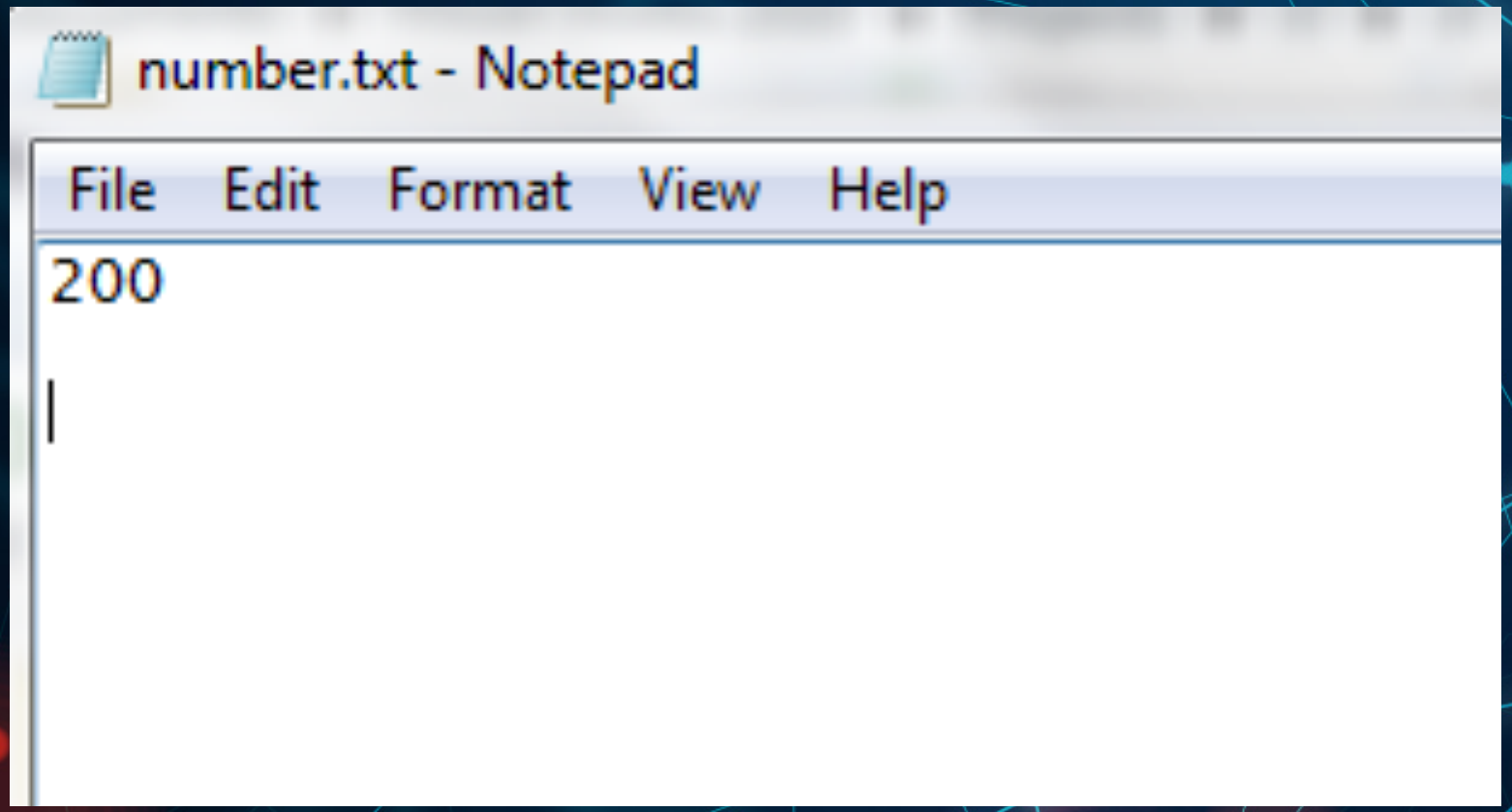
- `fsOut.open(const char[] fname)`
  - connects stream `fsOut` to the external file *`fname`*.
- `fsOut.put(char c)`
  - inserts character *`c`* to the output stream `fsOut`.
- `fsOut.eof()`
  - tests for the end-of-file condition.
- `fsOut.close()`
  - disconnects the stream and associated file.
- `fsOut << c;` //Behaves just like cout

File format

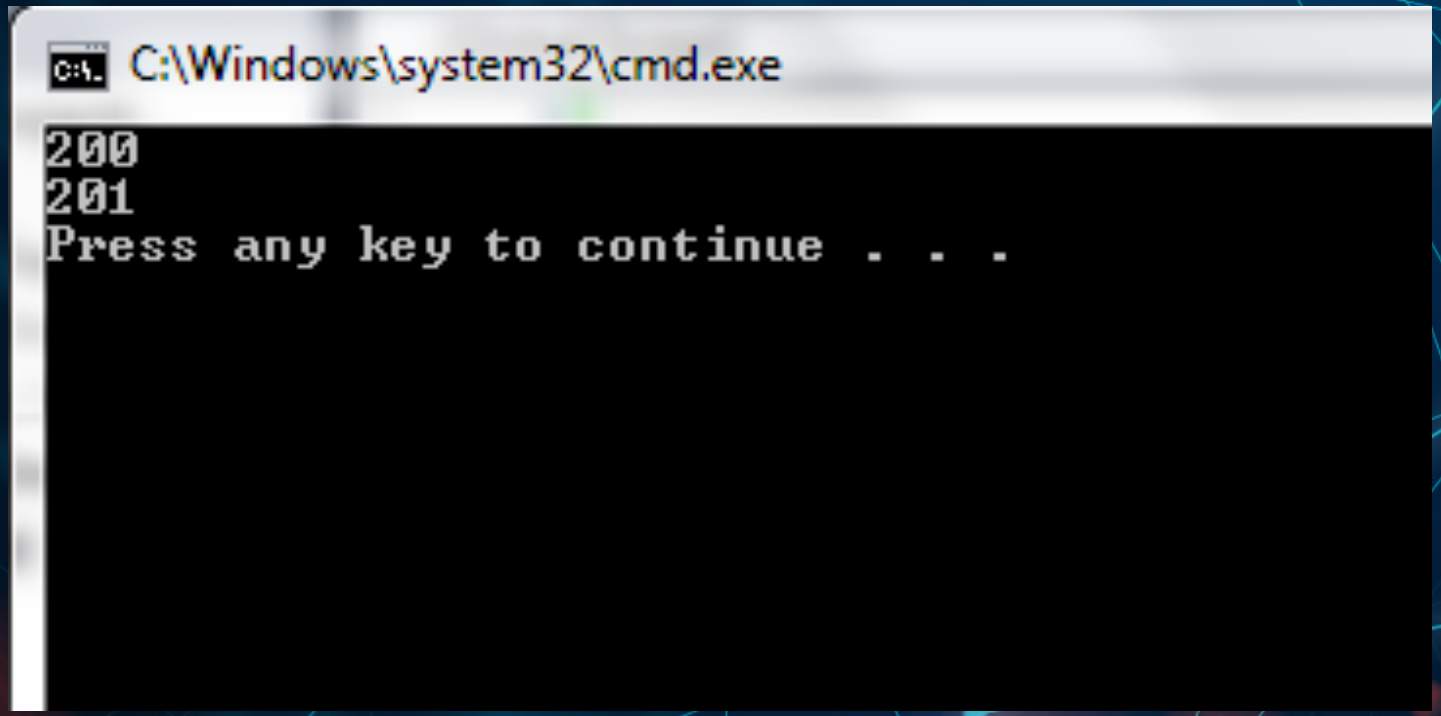
- In c++ files we (read from/ write to) them as a stream of characters
- What if I want to write or read numbers ?

Example writing to file

```
#include <iostream>
#include <fstream>
using namespace std;
void  main()
{
     ofstream outFile;
// open an exist file fout.txt
     outFile.open("number.txt",ios::app);

     if (!outFile.is_open())
     {
          cout << " problem with opening the file ";
     }
     else
     {
          outFile <<200 <<endl ;
          cout << "done writing" <<endl;
          outFile.close();
     }
}
```



```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
using namespace std;
void main()
{//Declare and open a text file
      ifstream INFile("number.txt");
      string line;
      int total=0;
      while(! INFile.eof())
      {
          getline(INFile, line);
          //converting line string to int
          stringstream(line) >> total;
          cout << line  <<endl;
          cout <<total +1<<endl;
      }
      INFile.close(); // close the file
}
```



Stream classes

Filebuf :- its purpose is to set the file buffer to read and write . Contain **openprot** constant used in the **open()** of file stream classes . Also contain **close()** and **open()** as method.

Fstreambase :- provides operations common to the file stream. Serves as a base for fstream, ifstream and ofsteram class. Contains **open()** and **close()** function

Ifstream :- provides input operations. Contains open() with default input mode. Inherits the functions **get()**, **getline()**, **read()**, **seekg()** and **tellg()** function from istream.

Ofstream :- provides output operations. Contains **open()** with default output mode. Inherits **put()**, **seekp()**, **teelp()** and **write()** function from ostream.

Fstream :- provides support for simultaneous input and output operations. Contains **open()** with default input mode. Inherits all the function from **istream** and ostream classes through **iostream**

Opening file using open()

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

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

Open and close a file

eg:-

```
ofstream  outfile;           // create stream
outfile . open (“DATA1”);  // connect stream to DATA1
……..
……..
outfile . Close();           //disconnect stream from DATA1
outfile . Open(“DATA2”); //connect stream to DATA2
……..
……..
outfile . close();
……..
```

Mode of file opening

ios :: out = open file for write only
ios :: in = open file for read only
ios :: app = append to end-of-file
ios :: ate = take us to the end of the file when it is opened

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 |)

eg :-

```
fstream file;
file . Open(“ data . txt”, ios :: out | ios :: in);
```

File pointer

Each file have two associated pointers known as the file pointers. One of them is called the input pointer (or get pointer) and the other is called the output pointer (or put pointer). The input pointer is used for reading the contents of a given file location and the output pointer is used for writing to a given file location.

Function for manipulation of file pointer

When we want to move file pointer to desired position then use these function for manage the file pointers.

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
```

put() and get() function

The function put() write a single character to the associated stream. Similarly, the function get() reads a single character from the associated stream.

read() and write() function

```
file . read ((char *)&V , sizeof (V));
file . Write ((char *)&V , sizeof (V));
```

These function take two arguments. The first is the address of the variable V , and the second is the length of that variable in bytes . The address of variable must be cast to type char * (i.e pointer to character type) .

Template Functions

```
#include <iostream>
using namespace

template <typename T>
T myMax(T x, T y)
{
   return (x > y)? x: y;
}

int main()
{
  cout << myMax<int>(3, 7) << endl;  // Call myMax for int
  cout << myMax<double>(3.0, 7.0) << endl; // call myMax for double
  cout << myMax<char>('g', 'e') << endl;   // call myMax for char

  return 0;
}
```

https://www.geeksforgeeks.org/templates-cpp/

Template Functions

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

template <class T>
void bubbleSort(T a[], int n) {
    for (int i = 0; i < n - 1; i++)
        for (int j = n - 1; i < j; j--)
            if (a[j] < a[j - 1])
            swap(a[j], a[j - 1]);
}

int main() {
    int a[5] = {10, 50, 30, 40, 20};
    int n = sizeof(a) / sizeof(a[0]);

    // calls template function
    bubbleSort<int>(a, n);

    cout << " Sorted array : ";
    for (int i = 0; i < n; i++)
        cout << a[i] << " ";
    cout << endl;
    return 0;
}
```

https://www.geeksforgeeks.org/templates-cpp/

WHAT ARE STRUCTS?

Struct

```
struct Person
{
    char name[50];
    int age;
    float salary;
};
```

https://www.programiz.com/cpp-programming/structure

Struct

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

struct Person
{
    char name[50];
    int age;
    float salary;
};

int main()
{
    Person p1;

    cout << "Enter Full name: ";
    cin.get(p1.name, 50);
    cout << "Enter age: ";
    cin >> p1.age;
    cout << "Enter salary: ";
    cin >> p1.salary;

    cout << "\nDisplaying Information." << endl;
    cout << "Name: " << p1.name << endl;
    cout <<"Age: " << p1.age << endl;
    cout << "Salary: " << p1.salary;

    return 0;
}
```

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
struct Representative  // Defining struct Representative
{
  string name;         // Name of a representative.
  double sales;        // Sales per month.
};
inline void print( const Representative& v)
{
   cout << fixed << setprecision(2)
        << left  << setw(20) << v.name
        << right << setw(10) << v.sales << endl;
}
int main()
{
   Representative rita, john;
   rita.name    = "Strom, Rita";
   rita.sales   = 37000.37;
   john.name    = "Quick, John";
   john.sales = 23001.23;

   rita.sales += 1700.11;              // More Sales
   cout << "  Representative             Sales\n"
        << "------------------------------" << endl;
   print( rita);
   print( john);
   cout << "\nTotal of sales: "
        << rita.sales + john.sales << endl;
   Representative *ptr = &john;          // Pointer ptr.
                                         // Who gets the
   if( john.sales < rita.sales)          // most sales?
     ptr = &rita;
   cout << "\nSalesman of the month: "
        << ptr->name << endl;    // Representative's name
                                 // pointed to by ptr.
   return 0;
}
```

STRUCTS

Image Reference : A Complete Guide to Programming in C++

STRUCTS

```
struct tm
{
   int tm_sec;          // 0 - 59(60)
   int tm_min;          // 0 - 59
   int tm_hour;         // 0 - 23
   int tm_mday;         // Day of month: 1 - 31
   int tm_mon;          // Month: 0 - 11 (January == 0)
   int tm_year;         // Years since 1900 (Year - 1900)
   int tm_wday;         // Weekday: 0 - 6 (Sunday == 0)
   int tm_yday;         // Day of year: 0 - 365
   int tm_isdst;        // Flag for summer-time
};
```

Image Reference : A Complete Guide to Programming in C++

STRUCTS

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

   struct tm *ptr;            // Pointer to struct tm.
   time_t sec;                // For seconds.
    . . .
   time(&sec);                // To get the present time.
   ptr = localtime(&sec);     // To initialize a struct of
                              // type tm and return a
                              // pointer to it.

   cout << "Today is the "       << ptr->tm_yday + 1
        << ". day of the year " << ptr->tm_year
        << endl;
    . . .
```