Day: Date: / /

Filing Basics

- Whenever we run a program, everything is stored on a ram and it is discarded when program stops but what if all of this stays saved?

  - For this we use files.

- For input output on terminal we required cout and cin as ~~object~~ object streams and we got them from <iostream>.

- For ~~file~~ input and output in a file we require file stream objects and we get them from:

```
#include <fstream>
```

File stream objects:

output file is used to write

1) of stream.
   output file stream.
   You can use this when you want to create and add data to a file.

input file is used to read.

2) if stream
   Input file stream
   You can use this to open an existing file and read data from it.

Yousaf

Day:

3) fstream

File stream. Objects of this data type can be used to open files for reading, writing or both. This will be covered later.

Creating a file object and opening a file:

- Before data can be written and read:
  - ✱ The file stream object must be created
  - ✱ The file must be opened and linked to the file stream object.

1) For writing:

```
fstream outputfile;
outputfile.open("test.txt");
```

(fstream: data type; outputfile: name of object; ofstream: data type; outputfile: name of file; open: opening; "test.txt": name of file.)

2) For reading

```
ifstream inputfile;
inputfile.open("test.txt");
```

Day: Date: / /

general:

1 file stream objects. identifier (name)
2 identifier . open ("name")

* identifier can be any name e.g input or file, you can also name an output file as input but don't do that.
* Second line opens the file that we want to open.

* Now the identifier will be used whereever we are reading or writing in the file, not the original file name.

Closing a file:

- Whenever a program terminates a file is automatically closed but it is always a good practice to close it.
  HOW?

```
identifier.close();
```

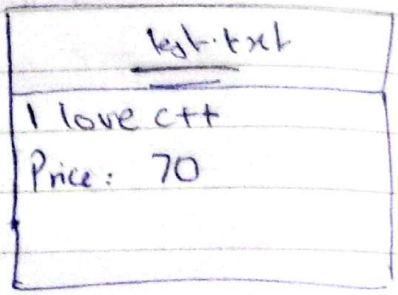
↑ no need to write the name.

Yousaf

Day: Date: / /

Writing data to file:

```
1  ofstream      outputfile;
2  outputfile.open("test.txt");
3
4  outputfile << "I love C++" << endl;
5  outputfile << "Price: " << price << endl;
6  outputfile.close();
```



This works just like cout but instead of outputting to a terminal you output to a file.

Important points:

- Whenever we open a file to write a new file is created and if a previous existed it will be deleted.

* So whenever we open a file it will be empty as it is newly created.

- If the file didn't exist it will be created.
- even if we don't write anything the file will be recreated and emptied if we open it for writing.

Yousuf

Day: Date: / /

- e.g

```
1   output file.open ("text.txt")
2   cout << "Hello" << endl;
3   output file.close()
4
5   output file.open ("text.txt")
6   cout << "Hi" << endl;
7   output file.close ()
```

~~This will~~ The file will only have Hi in it as the file was reopend. so re created.

Reading data from a file:

- Reading Data is very different.

e.g when we use cin we can get one number of like a single (not even a whole line). all of these problem also align with files.

Day: Date:

- if we are reading like a single number of a single word then simply:

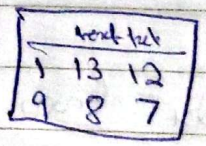
```
inputs file >> number;
cout << number;
```

or

```
input file >> line;
cout << line;
```

1) For many number in a file we can use a loop which is what we used if we need many numbers during cin.

ie




```
1 int number;
2 while (inputfile >>number){
3         cout << number;
4     }
```

The while loop will run until there is a number, if no number then it end.

Day: Date: / /

2) But what about a whole line

when we used to cin a whole line we used getline so the same will be used here.

```
1    string line;
2    while ( getline ( inputfile, line)){
3          cout << line << endl;
4     }
```

This will give you lines.

3) what if we want characters.

Then we can change:

```
1    char ch;
2    while ( inputfile >> ch ){
3          cout << ch<<endl;
4     }
```

This gives characters.

Day: Date:

Using EOF():

eof();

end of file.

This function tells us about the end of file.

e.g to read a file we can use

```
while (e !.eof()){
    getline(inputfile, line);
        cout << line << endl;
    }
```

Usage:

This will be used later e.g when we try to append a file instead of overwriting it.

We can goto eof and start writing from there.

Day: Date: / /

Precautions

Testing the file for errors:

File can give errors when we open them for reading if doesn't exist. This will be a logical error and not a syntax error.

How to check?

1)

This returns true if file is opened and false if file could not be open.

```
if (input file) {
    body.
}
else {
    cout << " Error: Could not open."
```

2)

```
if ( ! inputfile) {
    cout << "Error" << endl;
}
else {
    & body
}
```