Stream States

- good
- fail
- bad
- eof
- clear
Many types of streams
OK, now you're not swiping at salmon. Now you're just splashing me.
“And, as you can see, we've got a stream of hot air coming from this direction.”
Stream Characteristics

- Streams flow in one direction
Stream Characteristics

- Program can pull items out of **input streams**
  - `cin >> x;`

- Extraction operator
  - `>>`
Stream Characteristics

- Program can put items into **output streams**
  - `cout << "output: " << x << endl;`

- Insertion operator
  - `<<`
Can have: Multiple Input/Output Streams

- **input**
- **executing program**
- **output data**
Stream Fail State

- Possible reasons for entering fail state include:
  - wrong datatype (eg, expect int and get an alpha)
Stream Fail State

- Possible reasons for entering fail state include:
  - wrong datatype (eg, expect int and get an alpha)

```cpp
int x = 0, y = 0;
cin >> x >> y;
cout << "x is: " << x
    << "y is: " << y << endl;
```

Input:
```
A
5
```
Stream Fail State

- Possible reasons for entering fail state include:
  - wrong datatype (e.g., expect int and get an alpha)

```cpp
int x = 0, y = 0;
cin >> x >> y;
cout << "x is: " << x << "y is: " << y << endl;
```

**input**
A
5

**output**
x is 0
y is 0
Possible reasons for entering fail state include:

- **Wrong datatype** (e.g., expect int and get an alpha)

```
int x = 0, y = 0;
cin >> x >> y;
cout << "x is: " << x << "y is: " << y << endl;
```

Input:
```
A 5
```

`cin` went to a "fail" state.
Stream Fail State

- Possible reasons for entering fail state include:
  - **wrong datatype** (eg, expect int and get an alpha)

```cpp
int x = 0, y = 0;
cin >> x >> y;
cout << "x is: " << x << "y is: " << y << endl;
```

**input**
A
5

```
x is 0
y is 0
```

NO reading took place
Stream Fail State

- When a stream enters the fail state,
  - Further I/O operations on that stream have no effect
  - Program does NOT halt or give an error message
I/O states

- Consists of 3 bits

<table>
<thead>
<tr>
<th>fail</th>
<th>bad</th>
<th>eof</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>
I/O states

- Consists of 3 bits

fail: T
bad: T
eof: F

bad() – true if badbit is set
I/O states

- Consists of 3 bits

fail | bad | eof
--- | --- | ---
T | F | T

`eof()` – true if `eofbit` is set
**I/O states**

- Consists of 3 bits

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>F</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>fail</td>
<td>bad</td>
<td>eof</td>
<td></td>
</tr>
</tbody>
</table>

**fail()** – true if badbit or failbit or eofbit is set
I/O states

- Consists of 3 bits

- good() – true if no bits set
What they mean

- **fail**: true if read failed
- **bad**: true if stream is corrupt
- **eof**: doesn’t react same way on all systems
  - **set**: when end-of-file char read
  - **unset**: on next read operation
  - **unset**: on cin.clear()
NEVER use `eof()` as check in a loop

NEVER do: `while(!cin.eof())`
NEVER use `eof()` as check in a loop

NEVER do: `while(!cin.eof())`

unless you want a major headache
while ( cin.fail() ) …

while ( !cin.good() ) …

while ( !cin ) …

All the above are equivalent
Checking stream fail state

If given a stream 'cin', the following are all equivalent:

```cpp
if ( cin.fail() ) ...
if ( !cin.good() ) ...
if ( !cin ) ...
```

All the above are equivalent
Checking stream fail state

\[ \text{cin} \gg x; \text{if} ( \text{cin.fail}() ) \ldots \]

\[ \text{cin} \gg x; \text{if} ( !\text{cin.good}() ) \ldots \]

\[ \text{cin} \gg x; \text{if} ( !\text{cin} ) \ldots \]

All the above are equivalent
Checking stream fail state

```cpp
cin >> x; if ( cin.fail() ) ...
cin >> x; if ( !cin.good() ) ...
cin >> x; if ( !cin ) ...
if ( !(cin >> x) ) ...
```

All the above are equivalent
Checking stream fail state

\begin{verbatim}
cin >> x; if ( cin.fail() ) ... 
cin >> x; if ( !cin.good() ) ... 
cin >> x; if ( !cin ) ... 
    if ( !(cin >> x) ) ... 
\end{verbatim}

All the above are equivalent
Checking stream fail state

`cin >> x; if ( !cin ) ...`

`if ( !(cin >> x) ) ...`

All the above are equivalent
Checking stream fail state

cin >> x; if ( !cin ) ...

if ( !(cin >> x) ) ...

Similar to:

x=k; if ( !x ) ...

if ( !(x=k) ) ...
The above two are equivalent
cin >> x;
while ( !cin ) {
    ... 
    cin >> x;
}

while ( !cin ) {
    ... 
    cin >> x;
}

The above two are equivalent
Once stream "fails"
NO reading every takes place again

Until

```cpp
    cin.clear();
```
```cpp
int getValue(string prompt)
{
    int value;
    string junk;
    cout << prompt;
    cin >> value;
    while (cin.fail()) {
        cin.clear();
        getline(cin, junk);
        cout << prompt;
        cin >> value;
    }
    return value;
}
```
```cpp
int getValue(string prompt) {
    int value;
    string junk;
    cout << prompt;
    cin >> value;
    while (cin.fail()) {
        cin.clear();
        getline(cin, junk);
        cout << prompt;
        cin >> value;
    }
    return value;
}
```
Example

```cpp
int getValue(string prompt)
{
    int value;
    string junk;
    cout << prompt;
    cin >> value;
    while ( cin.fail() ) {
        cin.clear();
        getline(cin, junk);
        cout << prompt;
        cin >> value;
    }
    return value;
}
```
int getValue(string prompt)
{
    int value;
    string junk;
    cout << prompt;
    cin >> value;
    while ( cin.fail() ) {
        cin.clear();

        cout << prompt;
        cin >> value;
    }
    return value;
}
int getValue(string prompt)
{
    int value;
    string junk;
    cout << prompt;
    cin >> value;
    while ( cin.fail() ) {
        cin.clear();
        getline(cin, junk);
        cout << prompt;
        cin >> value;
    }
    return value;
}
int getValue(string prompt)
{
    int value;
    string junk;
    cout << prompt;
    cin >> value;
    while ( cin.fail() ) {
        cin.clear();
        getline(cin, junk);
        cout << prompt;
        cin >> value;
    }
    return value;
}
Example

```cpp
int getValue(string prompt)
{
    int value;
    string junk;
    cout << prompt;
    cin >> value;
    while ( !cin ) {
        cin.clear();
        getline(cin, junk);
        cout << prompt;
        cin >> value;
    }
    return value;
}
```
Example

```cpp
int getValue(string prompt)
{
    int value;
    string junk;
    cout << prompt;

    while ( !(cin >> value) ) {
        cin.clear();
        getline(cin, junk);
        cout << prompt;
    }
    return value;
}
```
File I/O

- fstream
- opening a file
- reading from file
- what happens when you hit the end of file
- stream states
Two main types of files: Text

- **Text file:** comprised of ASCII characters

Files you type with:
MS Visual pico, simpletext, notepad, wordpad
Two main types of files: Binary

- **Binary file**: structure/contents determined by program which wrote it

- **MSWord files**: jpg, mp3
```cpp
#include <iostream>

using namespace std;

int main()
{
    char ch;
    ch = 'X';
    cout << ch;
    return 0;
}
```
Basic input/output

#include <iostream>

cin -> executing program -> cout

input

output
Multiple Input/Output Streams

executing program

input

output data
Multiple Input/Output Streams

#include <fstream>

executing program

input

output data

#include <fstream>
Reading from file: 5 steps

1) `#include <fstream>`
   contains data types
   `ifstream, ofstream, and`
   functions
   `open and close`
Reading from file: 5 steps

1) `#include <fstream>`

2) **Declare a file stream variable** to represent the file you intend to use

```cpp
#include <fstream>

#include <fstream>

int main() {
    ifstream ins;  // for input
    // code to read from the file
    return 0;
}
```
Reading from file: 5 steps

1) #include <fstream>

2) Declare a file stream variable to represent the file you intend to use

```cpp
ifstream ins;    // for input
ifstream inFile;
ifstream myFile;
```

datatype  
variable name
Reading from file: 5 steps

1) `#include <fstream>`

2) **Declare a file stream variable** to represent the file you intend to use

```cpp
#include <fstream>

ifstream ins; // for input
ofstream outFile; // for output
```
Reading from file: 5 steps

1) #include <fstream>

2) Declare a file stream variable to represent the file you intend to use

3) Open the file

```cpp
ins.open("data.txt");
outFile.open("output.txt");
```
Reading from file: 5 steps

1) #include <fstream>

2) Declare a file stream variable to represent the file you intend to use

3) Open the file

4) Use this variable (file stream) to read
   ins >> num;
Reading from file: 5 steps

1) `#include <fstream>`

2) Declare a file stream variable to represent the file you intend to use

3) Open the file

4) Use this variable (file stream) to write

```
outFile << num;
```
Reading from file: 5 steps

1) `#include <fstream>`

2) Declare a file stream variable to represent the file you intend to use

3) Open the file

4) Use this variable (file stream) to read or write

5) **Close the file** when done with it

```
ins.close();
outFile.close();
```
Example

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

ifstream myInfile;  // declare stream variables
ofstream myOutfile;

myInfile.open("myIn.dat");  // open files
myOutfile.open("myOut.dat");

int x, y;  // use to read/write
myInfile >> x >> y;
myOutfile << x << y;

myInfile.close();  // close files
myOutfile.close();
```
inFile.open ( "data.txt" );

more on this in a couple more weeks
What does opening a file do?

```cpp
ifstream inFile;
inFile.open ( "data.txt" );
```

identifier within code
What does opening a file do?

```cpp
ifstream inFile;
inFile.open( "data.txt" );
```

**identifier within code**

**actual file (name of file on hard disk)**
What does opening a file do?

```cpp
ifstream inFile;
inFile.open( "data.txt" );
```

- `ifstream` is an identifier within code.
- `inFile` associates the identifier with the actual file (name of file on hard disk).
What does opening a file for input do?

```cpp
ifstream inFile;
inFile.open ( "data.txt" );
```

- Associates identifier for file with actual file

- If "data.txt" does **not** exist on disk or is NOT found
  - open is not successful

- Places `file reading marker` at beginning
What does opening a file do?

```cpp
ofstream outFile;
outFile.open("outputData.txt");
```

- **identifier within code**: `outFile`
- **associates the 2**
- **actual file (name of file on hard disk)**: `outputData.txt`
What does opening a file for output do?

```cpp
ofstream outFile;
outFile.open ( "data.txt" );
```

- Associates identifier for file with actual file
- If "data.txt " does **not** exist
  - new file with that name is created
- If "data.txt " **already** exists
  - it is erased
- Places *file writing marker* at beginning
#include <fstream>
#include <iostream>
using namespace std;

int main ()
{
    ifstream ins;
    ins.open("data.txt");
    int x;
    ins >> x;
    while ( ins.good() ) {
        cout << x;
        ins >> x;
    }
    ins.close();
    return 0;
}
#include <fstream>
#include <iostream>
using namespace std;

int main ()
{
    ifstream ins;
    ins.open("data.txt");
    int x;
    ins >> x;
    while ( ins.good() ) {
        cout << x;
        ins >> x;
    }
    ins.close();
    return 0;
}
#include <fstream>
#include <iostream>
using namespace std;

int main ( )
{
    ifstream ins;
    ins.open("data.txt");
    int x;
    ins >> x;
    while ( ins ) {
        cout << x;
        ins >> x;
    }
    ins.close();
    return 0;
}
Example: output all data in a file

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

int main ( )
{
    ifstream ins;
    ins.open("data.txt");
    int x;

    while ( ins >> x ) {
        cout << x;
    }

    ins.close();
    return 0;
}
```
#include <fstream>
#include <iostream>
using namespace std;

int main ()
{
    ifstream ins;
    ins.open("data.txt");
    int x;
    ins >> x;
    ins >> x;
    while (!ins.eof()) {
        cout << x;
        ins >> x;
    }
    ins.close();
    return 0;
}
More reads than data available

```c
int i=0, j=0;
char ch=' ', k=' ';
inFile >> i >> j >> ch >> k;
```
Common Error: read past `<eof>`

More reads than data available

```c
int i=0, j=0;
char ch=' ', k=' '; 
inFile >> i >> j >> ch >> k;
```
Common Error:
read past `<eof>`

More reads than data available

```cpp
int i=0, j=0;
char ch=' ', k=' ';
inFile >> i >> j >> ch >> k;
```
More reads than data available

```c
int i=0, j=0;
char ch=' ', k=' ';
inFile >> i >> j >> ch >> k;
```
Common Error: read past `<eof>`

More reads than data available

```c
int i=0, j=0;
char ch=' ', k=' ';  
inFile >> i >> j >> ch >> k;
```

no input left for 'k'
More reads than data available

```c
int i=0, j=0;
char ch=' ', k=' ';
inFile >> i >> j >> ch >> k;
inFile.fail() set
inFile.eof() set
```

Common Error: read past `<eof>`

15\-5Q

no input left for 'k'
More reads than data available

```cpp
int i=0, j=0;
char ch=' ', k=' ';
inFile >> i >> j >> ch >> k >> z;
```
Another Common Error: file does not open

- File name spelled incorrectly
  - inFile.open ( "data.text" );
  - not in the correct directory
- Effect
  - inFile would enter *fail state*;
  - all attempts to read from file would fail
Another Common Error: file does not open

- File name spelled incorrectly
  - inFile.open ("data.text");
  - not in the correct directory

- Effect
  - inFile would enter *fail state*;
  - all attempts to read from file would fail

- Program does not automatically halt the program nor give any error message
I/O states
Same for keyboard and files

- Consists of 3 bits

<table>
<thead>
<tr>
<th></th>
<th>fail</th>
<th>bad</th>
<th>eof</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>

- `bad()` — true if badbit is set
- `eof()` — true if eofbit is set
- `fail()` — true if any bit is set
- `good()` — true if no bits are set
ifstream inStream;

inStream.open("stuff.dat");

if ( inStream.fail() ) //check if opened
{
    cout << "Opening failed\n";
    return;
}
Read a file name and then open

- Two types of "strings": C++ standard library and C style

```cpp
string c++String;
```

C++ string
Read a file name and then open

- Two types of "strings": C++ standard library and C style

```cpp
string c++String = "cString";
```

C++ string  C string
Read a file name and then open

- Two types of "strings": C++ standard library and C style

```cpp
string c++String = "cString";
```

Automatic conversion from C to C++ string
Read a file name and then open

- Two types of "strings": C++ standard library and C style

```cpp
string fileName;
cout << "Enter file name: ";
cin >> fileName;
inFile.open(fileName.c_str());
```
Read a file name and then open

- Two types of "strings": C++ standard library and C style

```cpp
string fileName;
cout << "Enter file name: ";
cin >> fileName;
inFile.open ( );
```
Read a file name and then open

- Two types of "strings": C++ standard library and C style

```cpp
string fileName;
cout << "Enter file name: ";
cin >> fileName;
inFile.open ( );
```
Read a file name and then open

- Two types of "strings": C++ standard library and C style

```cpp
string fileName;
cout << "Enter file name: ";
cin >> fileName;
inFile.open ("data.txt");
```

Requires C string
Read a file name and then open

- Two types of "strings": C++ standard library and C style

```cpp
string fileName;
cout << "Enter file name: ";
cin >> fileName;
inFile.open ( fileName.c_str );
```

C++ string

Requires C string
Read a file name and then open

- Two types of "strings": C++ standard library and C style

```cpp
string fileName;
cout << "Enter file name: ";
cin >> fileName;
inFile.open ( fileName.c_str() );
```
Problem

• Read a datafile and count
• How many:
  • capital letters
  • lowercase letters
  • digits
  • punctuation
  • ?????????
void openFile (ifstream& ins) {
    string fileName;

    cout << "Enter the name of the file: ";
    cin >> fileName;

    ins.open(fileName.c_str());

    while (!ins) {
        ins.clear(); //ins is in a fail state
        cout << "Error in opening file " << endl;
        cin >> fileName;

        ins.open(fileName.c_str());
    }
}
void openFile (ifstream& ins)
{
    string fileName;

    cout << "Enter the name of the file: ";
    cin >> fileName;

    ins.open(fileName.c_str());

    while (!ins) {
        ins.clear();           //ins is in a fail state
        cout << "Error in opening file " << endl;
        cout << "Enter the name of the file: ";
        cin >> fileName;

        ins.open(fileName.c_str());
    }
}
```cpp
void openFile (ifstream& ins) {
    string fileName;

    cout << "Enter the name of the file: ";
    cin >> fileName;

    ins.open(fileName.c_str());

    while (!ins) {
        ins.clear();           // ins is in a fail state
        cout << "Error in opening file " << endl;
        cout << "Enter the name of the file: ";
        cin >> fileName;

        ins.open(fileName.c_str());
    }
}
```
void openFile (ifstream& ins)
{
    string fileName;

    cout << "Enter the name of the file: ";
    cin >> fileName;

    ins.open(fileName.c_str());

    while (!ins) {
        ins.clear();           //ins is in a fail state
        cout << "Error in opening file " << endl;
        cout << "Enter the name of the file: ";
        cin >> fileName;

        ins.open(fileName.c_str());
    }
}
void openFile (ifstream& ins)
{
    string fileName;

    cout << "Enter the name of the file: ";
    cin >> fileName;

    ins.open(fileName.c_str());

    while (!ins) {
        ins.clear(); //ins is in a fail state
        cout << "Error in opening file " << endl;
        cout << "Enter the name of the file: ";
        cin >> fileName;

        ins.open(fileName.c_str());
    }
}
```cpp
void openFile (ifstream& ins)
{
    string fileName;

    cout << "Enter the name of the file: ";
    cin >> fileName;

    ins.open(fileName.c_str());

    while (!ins) {
        ins.clear(); //ins is in a fail state
        cout << "Error in opening file " << endl;
        cout << "Enter the name of the file: ";
        cin >> fileName;

        ins.open(fileName.c_str());
    }
}
```
void openFile (ifstream& ins) {
    string fileName;

    cout << "Enter the name of the file: ";
    cin >> fileName;

    ins.open(fileName.c_str());

    while (!ins) {
        ins.clear(); //ins is in a fail state
        cout << "Error in opening file " << endl;
        cout << "Enter the name of the file: ";
        cin >> fileName;

        ins.open(fileName.c_str());
    }
}
void openFile (ifstream& ins) {
    string fileName;

    cout << "Enter the name of the file: ";
    cin >> fileName;

    ins.open(fileName.c_str());

    while (!ins) {
        ins.clear(); // ins is in a fail state
        cout << "Error in opening file " << endl;
        cout << "Enter the name of the file: ";
        cin >> fileName;

        ins.open(fileName.c_str());
    }
}
void openFile (ifstream& ins)
{
    string fileName;

    cout << "Enter the name of the file: ";
    cin >> fileName;

    ins.open(fileName.c_str());

    while (!ins) {
        ins.clear();       // ins is in a fail state
        cout << "Error in opening file " << endl;
        cout << "Enter the name of the file: ";
        cin >> fileName;

        ins.open(fileName.c_str());
    }
}
void openFile (ifstream& ins)
{
    string fileName;

    cout << "Enter the name of the file: ";
    cin >> fileName;

    ins.open(fileName.c_str());

    while (!ins) {
        ins.clear(); //ins is in a fail state
        cout << "Error in opening file " << endl;
        cout << "Enter the name of the file: ";
        cin >> fileName;
        ins.open(fileName.c_str());
    }
}

void openFile (ifstream& ins)
{
    string fileName;

    cout << "Enter the name of the file: ";
    cin >> fileName;

    ins.open(fileName.c_str());

    while (!ins) {
        ins.clear();          //ins is in a fail state
        cout << "Error in opening file " << endl;
        cout << "Enter the name of the file: ";
        cin >> fileName;

        ins.open(fileName.c_str());
    }
}

Problem

• Read a datafile and count

• How many:
  • capital letters
  • lowercase letters
  • digits
  • punctuation
  • ????????
Count Capital Letters

- Use ASCII table ordering

```java
if (ch >= 'A' && ch <= 'Z')
capitalCount++;
```
### Count Capital Letters

- Use ASCII table ordering

```c
if (ch >= 'A' && ch <= 'Z')
    capitalCount++;
```

- or use `cctype`

```c
if (isupper(ch))
    capitalCount++;
```
#include <cctype>

- `toupper(char)` returns upper case version
- `tolower(char)` returns lower case version
- `isupper(char)` true if upper
- `islower(char)` true if lower
- `isalpha(char)` true if letter
- `isdigit(char)` true if digit ‘0’ – ‘9’
- `isalnum(char)` true if either letter or digit
```c
#include <cctype>

- **isspace(ch)**  true if WS
- **ispunct(ch)**  true if printing char other than WS, digit, letter
- **isprint(ch)**  true if printing char
- **isgraph(ch)**  true if printing char other than WS
- **isctrl(ch)**   true if control char (ASCII 0-31 or 127)
```
Convert Char to Lowercase Letter if possible

```java
ch = tolower (ch);
```
What if we had to write the function itself?

- in ASCII, code for 'A' is 65, code for 'a' is 97
- in ASCII, code for 'B' is 66, code for 'b' is 98

etc.

so what would we do???
char myToLowerCase(char ch)
{
    if (( 'A' <= ch ) && ( ch <= 'Z' ))
        return ( char ( ch + 32 ));
    else
        return ( ch );
}
if (isdigit (ch))
    num = int (ch - '0');
else
    cout « "ERROR! not numeric";

If \texttt{ch} is '5', then \texttt{num} \leftarrow 53 - 48 = 5
Problem

• Read a datafile and count
• How many:
  • capital letters
  • lowercase letters
  • digits
  • punctuation
  • ?????????
int main()
{
    char ch;
    int lowerCount=0, upperCount=0, spaceCount=0, punctCount=0, cntrlCount=0, otherCount=0,
    digitCount=0;

    ifstream ins;
    openFile(ins);

    ins.get(ch);
    while (ins) {
        if (isdigit(ch))       digitCount++;
        else if (islower(ch)) lowerCount++;
        else if (isupper(ch)) upperCount++;
        else if (isspace(ch)) spaceCount++;
        else if (ispunct(ch)) punctCount++;
        else if (iscntrl(ch)) cntrlCount++;
        else otherCount++;

        ins.get(ch);
    }
    printResults(digitCount,lowerCount,upperCount,spaceCount,punctCount,cntrlCount,otherCount);

    ins.close();
    return 0;
}
int main()
{
    char ch;
    int lowerCount=0, upperCount=0, spaceCount=0, punctCount=0, cntrlCount=0, otherCount=0,
        digitCount=0;

    ifstream ins;
    openFile(ins);

    ins.get(ch);
    while (ins) {
        if (isdigit(ch))    digitCount++;
        else if (islower(ch)) lowerCount++;
        else if (isupper(ch)) upperCount++;
        else if (isspace(ch)) spaceCount++;
        else if (ispunct(ch)) punctCount++;
        else if (iscntrl(ch)) cntrlCount++;
        else otherCount++;

        ins.get(ch);
    }
    printResults(digitCount,lowerCount,upperCount,spaceCount,punctCount,cntrlCount,otherCount);

    ins.close();
    return 0;
}
int main()
{
    char ch;
    int lowerCount=0, upperCount=0, spaceCount=0, punctCount=0, cntrlCount=0, otherCount=0,
        digitCount=0;

    ifstream ins;
    openFile(ins);

    ins.get(ch);
    while (ins) {
        if (isdigit(ch))    digitCount++;
        else if (islower(ch)) lowerCount++;
        else if (isupper(ch)) upperCount++;
        else if (isspace(ch)) spaceCount++;
        else if (ispunct(ch)) punctCount++;
        else if (iscntrl(ch)) cntrlCount++;
        else otherCount++;

        ins.get(ch);
    }
    printResults(digitCount,lowerCount,upperCount,spaceCount,punctCount,cntrlCount,otherCount);

    ins.close();
    return 0;
}
int main()
{
    char ch;
    int lowerCount=0, upperCount=0, spaceCount=0, punctCount=0, cntrlCount=0, otherCount=0, digitCount=0;

    ifstream ins;
    openFile(ins);

    ins.get(ch);
    while (ins) {
        if (isdigit(ch))        digitCount++;
        else if (islower(ch))   lowerCount++;
        else if (isupper(ch))   upperCount++;
        else if (isspace(ch))   spaceCount++;
        else if (ispunct(ch))   punctCount++;
        else if (iscntrl(ch))   cntrlCount++;
        else otherCount++;

        ins.get(ch);
    }
    printResults(digitCount,lowerCount,upperCount,spaceCount,punctCount,cntrlCount,otherCount);

    ins.close();
    return 0;
}
int main()
{
    char ch;
    int lowerCount=0, upperCount=0, spaceCount=0, punctCount=0, cntrlCount=0, otherCount=0, digitCount=0;

    ifstream ins;
    openFile(ins);

    ins.get(ch);
    while (ins) {
        if (isdigit(ch))      digitCount++;
        else if (islower(ch)) lowerCount++;
        else if (isupper(ch)) upperCount++;
        else if (isspace(ch)) spaceCount++;
        else if (ispunct(ch)) punctCount++;
        else if (iscntrl(ch)) cntrlCount++;
        else otherCount++;

        ins.get(ch);
    }
    printResults(digitCount,lowerCount,upperCount,spaceCount,punctCount,cntrlCount,otherCount);

    ins.close();
    return 0;
}
int main()
{
    char ch;
    int lowerCount=0, upperCount=0, spaceCount=0, punctCount=0, cntrlCount=0, otherCount=0, digitCount=0;

    ifstream ins;
    openFile(ins);

    ins.get(ch);
    while (ins) {
        if (isdigit(ch))    digitCount++;
        else if (islower(ch)) lowerCount++;
        else if (isupper(ch)) upperCount++;
        else if (isspace(ch)) spaceCount++;
        else if (ispunct(ch)) punctCount++;
        else if (iscntrl(ch)) cntrlCount++;
        else otherCount++;

        ins.get(ch);
    }
    printResults(digitCount,lowerCount,upperCount,spaceCount,punctCount,cntrlCount,otherCount);

    ins.close();
    return 0;
}
int main()
{
    char ch;
    int lowerCount=0, upperCount=0, spaceCount=0, punctCount=0, cntrlCount=0, otherCount=0,
        digitCount=0;

    ifstream ins;
    openFile(ins);

    ins.get(ch);
    while (ins) {
        if (isdigit(ch))    digitCount++;
        else if (islower(ch)) lowerCount++;
        else if (isupper(ch)) upperCount++;
        else if (isspace(ch)) spaceCount++;
        else if (ispunct(ch)) punctCount++;
        else if (iscntrl(ch)) cntrlCount++;
        else otherCount++;

        ins.get(ch);
    }
    printResults(digitCount,lowerCount,upperCount,spaceCount,punctCount,cntrlCount,otherCount);

    ins.close();
    return 0;
}
```c
int main()
{
    char ch;
    int lowerCount=0, upperCount=0, spaceCount=0, punctCount=0, cntrlCount=0, otherCount=0, digitCount=0;

    ifstream ins;
    openFile(ins);

    ins.get(ch);
    while (ins) {
        if (isdigit(ch))     digitCount++;
        else if (islower(ch)) lowerCount++;
        else if (isupper(ch)) upperCount++;
        else if (isspace(ch)) spaceCount++;
        else if (ispunct(ch)) punctCount++;
        else if (iscntrl(ch)) cntrlCount++;
        else otherCount++;

        ins.get(ch);
    }
    printResults(digitCount,lowerCount,upperCount,spaceCount,punctCount,cntrlCount,otherCount);

    ins.close();
    return 0;
}
```
int main()
{
    char ch;
    int lowerCount=0, upperCount=0, spaceCount=0, punctCount=0, cntrlCount=0, otherCount=0,
        digitCount=0;

    ifstream ins;
    openFile(ins);

    while (ins.get(ch)) {
        if (isdigit(ch)) digitCount++;
        else if (islower(ch)) lowerCount++;
        else if (isupper(ch)) upperCount++;
        else if (isspace(ch)) spaceCount++;
        else if (ispunct(ch)) punctCount++;
        else if (iscntrl(ch)) cntrlCount++;
        else otherCount++;
    }

    printResults(digitCount,lowerCount,upperCount,spaceCount,punctCount,cntrlCount,otherCount);

    ins.close();
    return 0;
}
#include <iostream>

using namespace std;
#include <iostream>
#include <fstream>
#include <string>
#include <cctype>

using namespace std;
Summary

- stream states
- clear()
- files
  - open
  - reading/writing
  - checking states