PROCESSING FILES

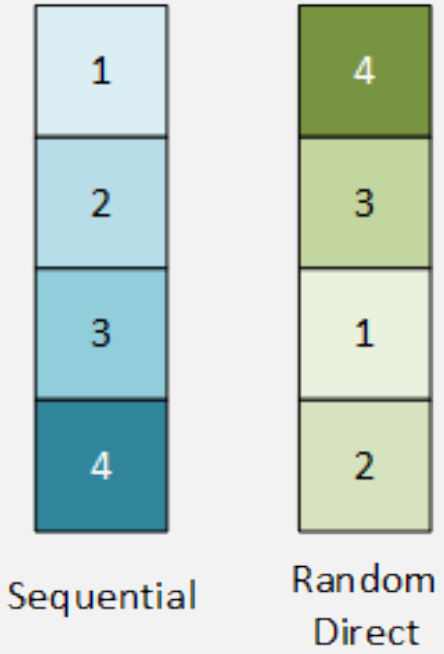
Access order and size

Control: `eof()`, `EOF`, and `operator bool`

Delroy A. Brinkerhoff

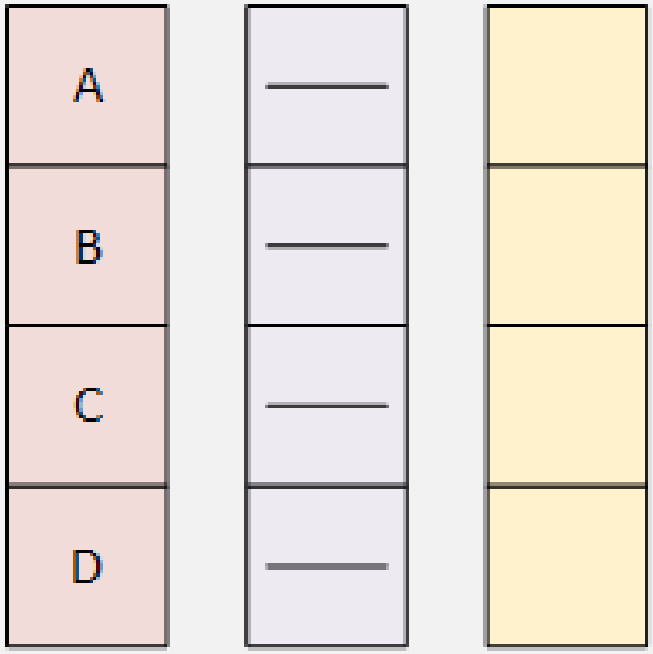
FILE PROCESSING ORDER

- Sequential
  - Reads or writes data from beginning to end
- Random / Direct
  - Read or write data in any order
  - Access data by address
- Keyed / Indexed
  - Requires a key or index file



FILE PROCESSING DATA SIZE

- Physical access in fixed-sized blocks
  - Operating system ↔ hardware storage
- Logical access in convenient units
  - Characters or bytes
  - Lines
  - Logical blocks



FILE PROCESSING SUMMARY

Size \ Order	Sequential	Random
Character	Text & Binary	
Line	Text	
Block	Binary (infrequently)	Binary
Buffer	Text & Binary	

THE eof FUNCTION

INCORRECT

```
ifstream file(file_name);

while (! file.eof())  // 1
{
    // read file      // 2
    // process data   // 3
}
```

CORRECT

```
ifstream file(file_name);

// initial file read  // 1
while (! file.eof())  // 2
{
    // process data   // 3
    // read file      // 4
}
```

SEQUENTIALLY READING CHARACTERS

```
int main()
{
    ifstream in("data.txt");
    char c;

    in >> c;
    while (!in.eof())
    {
        cout << '|' << c << '|' << endl;
        in >> c;
    }

    return 0;
}
```

- `istream& operator>>(int& c);`
- `istream& get(char& c);`
- `int get();`

CHARACTER INPUT AND EOF

```
int main()
{
    ifstream in("data.txt");
    int c;

    while ((c = in.get()) != EOF)
        cout << (char)c << endl;

    return 0;
}
```

operator bool

- operator bool
  - Overloaded casting operator
  - Stream → Boolean
  - failbit set if an I/O operation fails
  - badbit set if a stream is corrupted
  - The conversion operator returns true if either flag is set, false otherwise
  - Returns: !(failbit | badbit)

```
int main()
{
    ifstream in("data.txt");
    char c;

    while (in.get(c))
        cout << '|' << c << '|' << endl;

    return 0;
}
```