CSC127 IO Stream Library

Introduction

- In C++ I/O system operates through **streams.**

- I/O system provides a level of **abstraction** between the **programmer** and the **device**.

- This **abstraction** is called a ***stream*** *and the* ***actual device*** *is called a* ***file.***

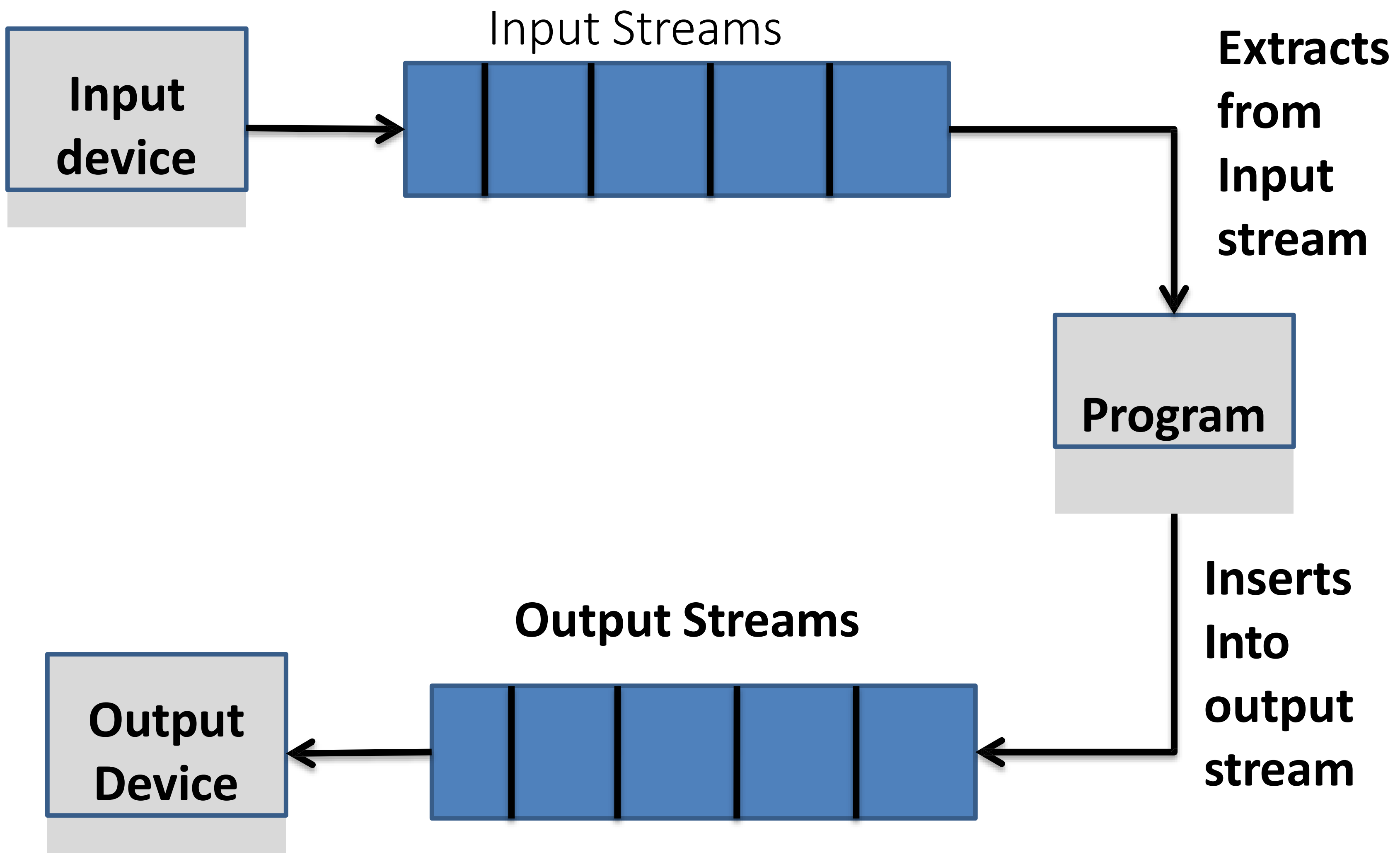
Introduction

- A stream is a **logical device** that either produces or consumes information.

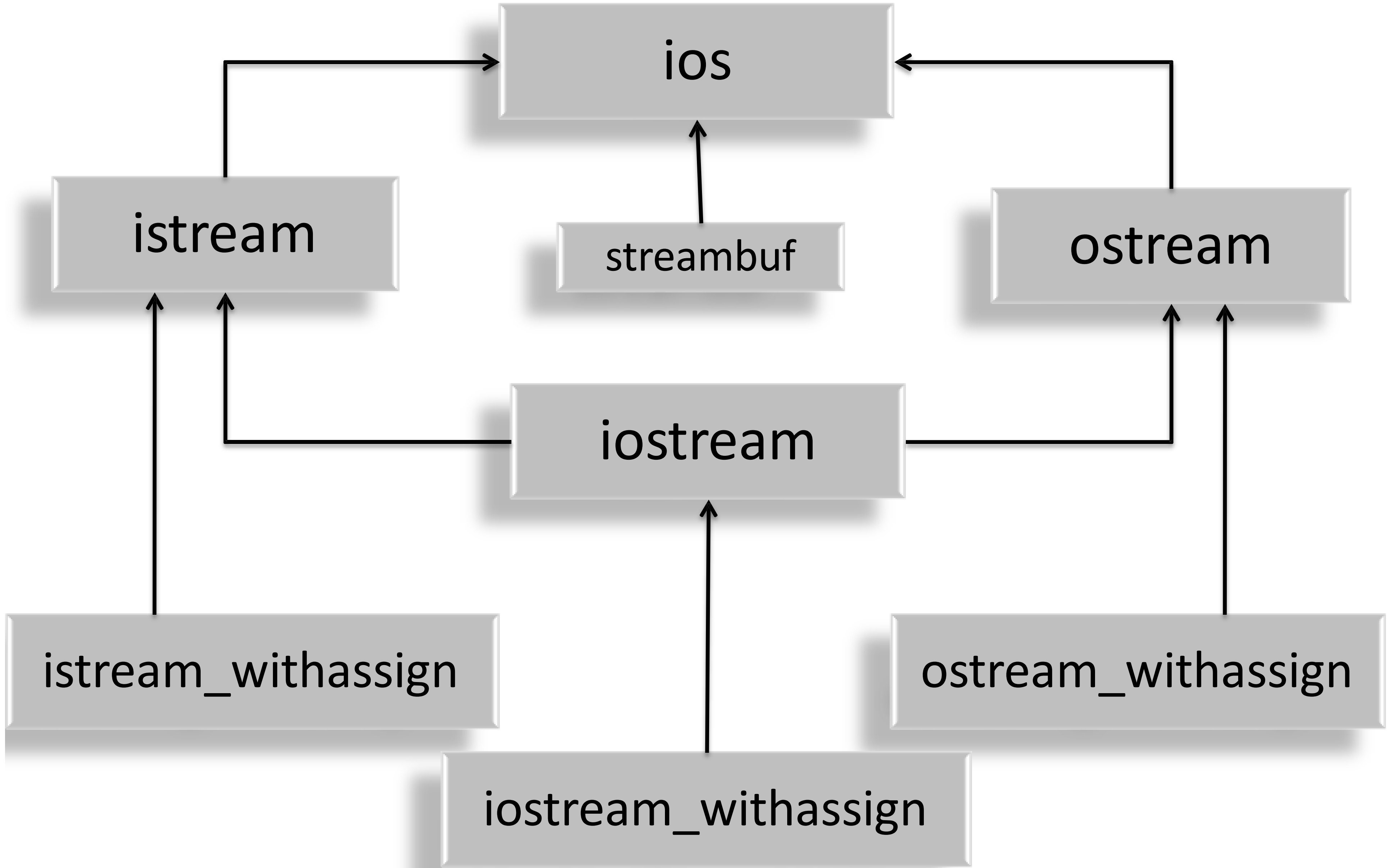
- A stream is **linked** to a **physical device** by the I/O system.

- Standard C++ provides support for its I/O system in **<iostream.h>**

C++ Streams


Input Streams
Input device
Extracts from Input stream
Program
Inserts Into output stream
Output Streams
Output Device

I/O Stream Classes for console Operations



C++'s Predefined Streams

When a C++ program begins execution, four built-in streams are automatically opened.

Stream	Meaning	Default Device
cin	Standard input	Keyboard
cout	Standard output	Screen

Unformatted I/O

- Input operator
- Output operator
- Overloading I/O Operator

Input Operator

- Extraction operator:(>>)
- float var;

  cin >>var;

  char line[20];

  cin>>line;

- get(), getline(),read()

Output Operator

- Insertion Operator:(<<)
- float var;

  char line[20];

  cout<< var<<line;

- put(),putline(),write()

Overloading >> operator

- **Prototype:**

  friend istream& operator >>(istream&, Matrix&);

- **Example:**

```
istream& operator >>(istream&
  in, Matrix& m)
{
    for(int i=0; i<row*col; i++)
    {
            in >> Mat[i];
     }
     return in;
}
```

```
void main()
{
   :
Cin>>mobj;
   :
}
```

Overloading << operator

- **Prototype:**

  friend ostream& operator <<(ostream&, Matrix&);

- ***Example:***

```
ostream& operator <<(ostream&
  out, Matrix& m)
{
      for(int i=0; i<row; i++)
{
        for(int j=0; j<col; j++)
        {   out>> Mat[i][j] >> "\t"
  ;   }
        out << endl;
```

```
void main()
{
 :
cout<<mobj;
 :
}
```

Formatted I/O

- There are three related but conceptually different ways that we can format data.

  - directly accessing members of the **ios** class.
  - using special functions called **manipulators**.
  - user defined output functions

Formatting Using the ios Members

- The **ios** class declares a bitmask enumeration called **fmtflags** in which the following set of format flags are defined.

- To set a flag, the setf() function is used. This function is a member of ios.

- **Syntax:** fmtflags setf(fmtflags flags);

  example: stream.setf(ios::showpos);

Flag	Meaning
skipws	leading white-space characters are discarded when performing input on a stream
left	output is left justified.
right	output is right justified. Default is right justified.
internal	a numeric value is padded to fill a field by inserting spaces between any sign or base character.
oct	flag causes output to be displayed in octal.
hex	flag causes output to be displayed in hexadecimal.
dec	flag causes output to be displayed in decimal. Default is decimal output.
showbase	Shows the base of numeric values

Flag	Meaning
showpos	causes a leading plus sign to be displayed before positive values.
scientific	floating-point numeric values are displayed using scientific notation. By default, when scientific notation is displayed, the e is in lowercase.
uppercase	characters are displayed in uppercase.
showpoint	causes a decimal point and trailing zeros to be displayed for all floating-point output
fixed	floating-point values are displayed using normal notation.
unitbuf	the buffer is flushed after each insertion operation.
boolalpha	Booleans can be input or output using the keywords true and false.

Function	Meaning
width( )	To specify required field size for displaying an output value.
precision( )	To specify the number of digits to displayed after the decimal point of a float value value.
fill( )	To specify a character to used to fill the unused portion of a field.
setf( )	Sets the format flags
unsetf( )	Un-Sets the format flags

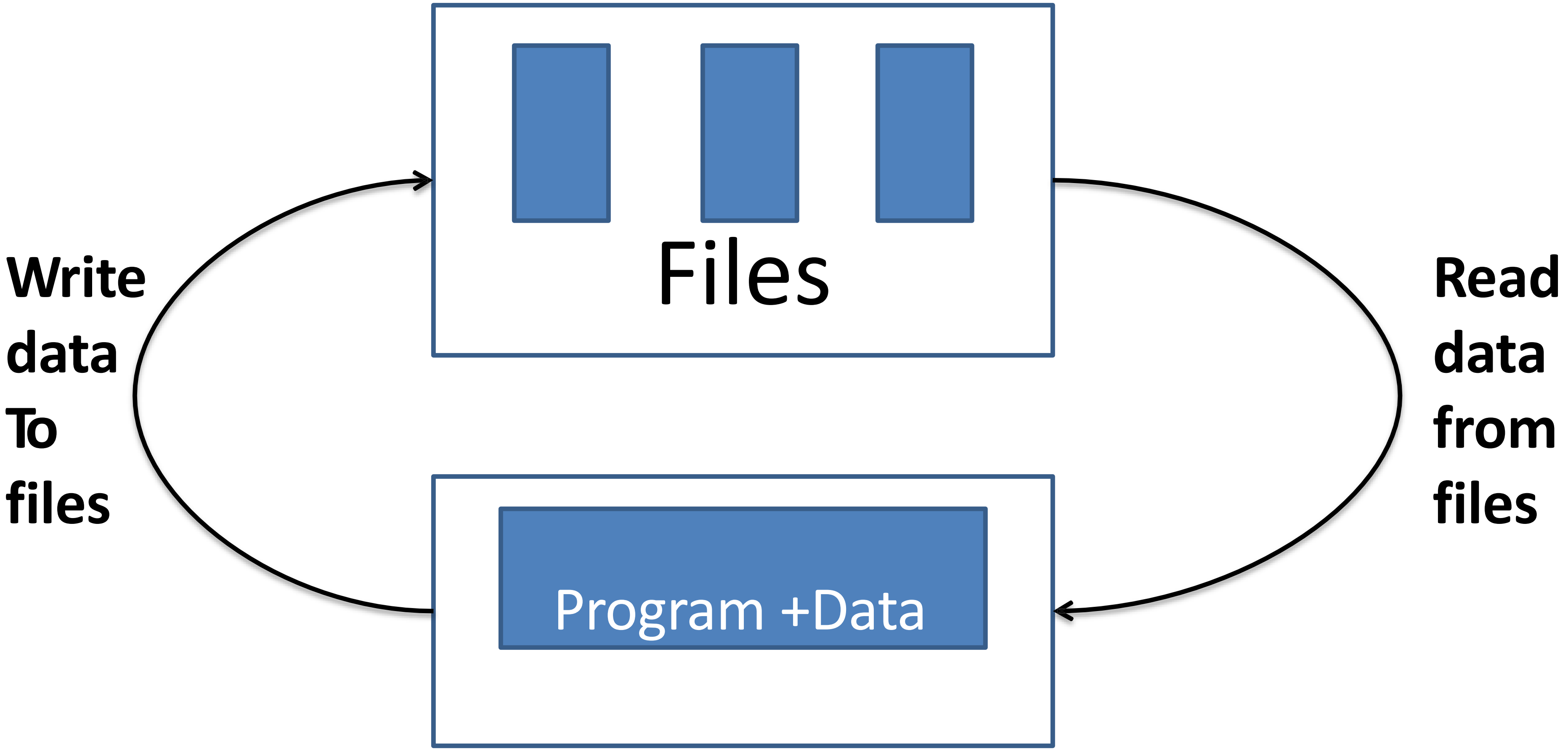
Using Manipulators to Format I/O

Manipulators	Meaning
boolalpha	Turns on **boolapha flag.**
dec	Turns on **dec flag.**
endl	Output a newline character and flush the stream.
ends	Output a null.
fixed	Turns on **fixed flag.**
flush	Flush a stream.
hex	Turns on **hex flag.**
internal	Turns on **internal flag.**
left	Turns on **left flag.**
noboolalpha	**Turns off boolalpha flag.**

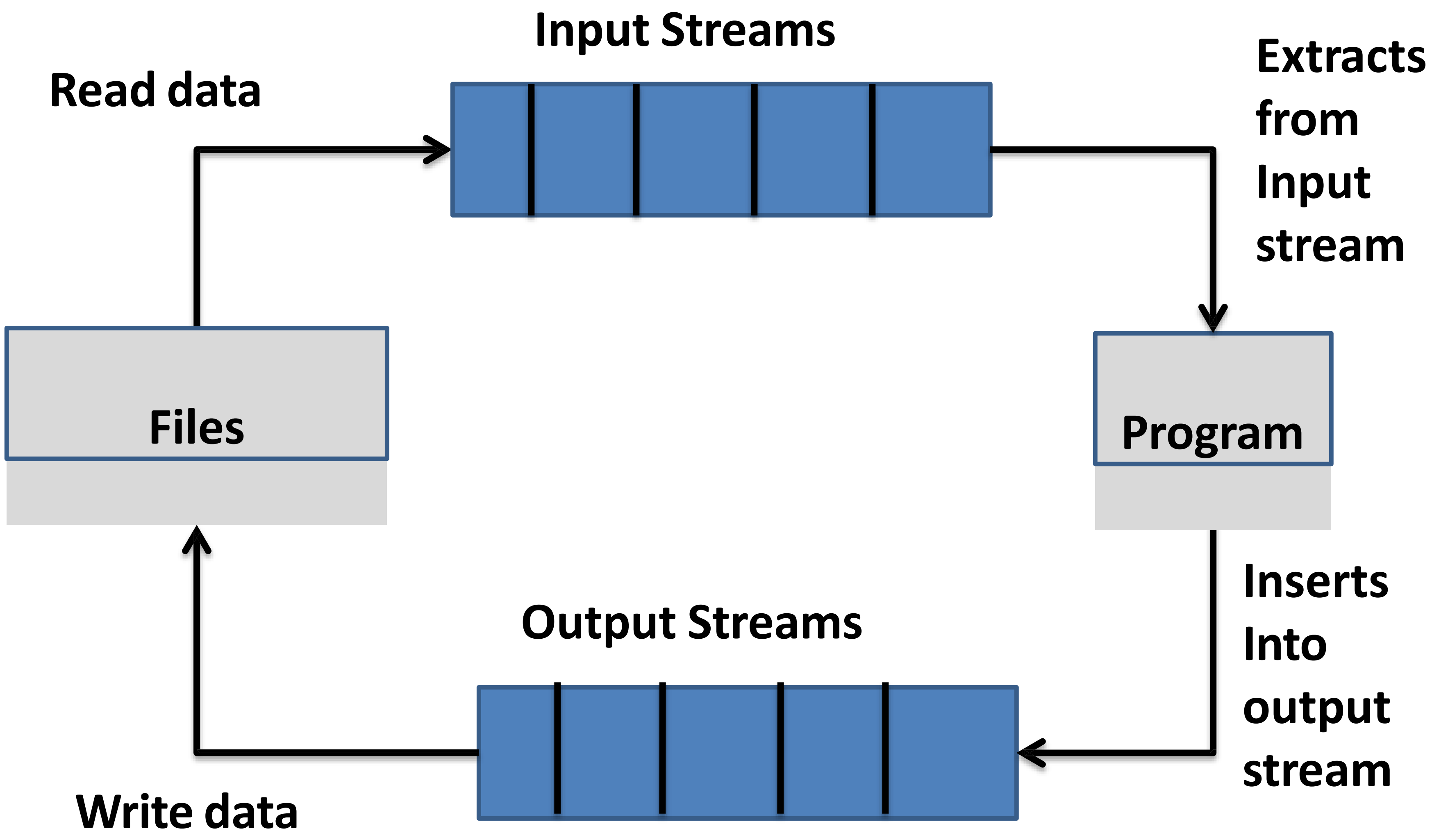
Manipulators	Meaning
noshowbase	Turns off **showbase flag.**
noshowpoint	Turns off **showpoint flag.**
no showpos	Turns off **showpos flag.**
noskipws	Turns off **skipws flag.**
nounitbuf	Turns off **unitbuf flag.**
nouppercase	Turns off **uppercase flag.**
oct	Turns on **oct flag.**
right	Turns on **right flag.**
scientfic	Turns on **scientific flag.**
setbase(int base)	**Set the number base to *base.***

Manipulators	Meaning
setfill(int ch)	Set the fill character to *ch.*
setiosflags(fmtflags f)	Turn on the flags specified in *f.*
setprecision(int p)	Set the number of digits of precision.
setw(int w)	Set the field width to *w.*
showbase	Turns on **showbase flag.**
showpoint	Turns on **showpoint flag.**
showpos	Turns on **showpos flag.**
skipws	Turns on **skipws flag.**
unitbuf	Turns on **unitbuf flag.**
uppercase	Turns on **uppercase flag.**
ws	**Skip leading white space.**

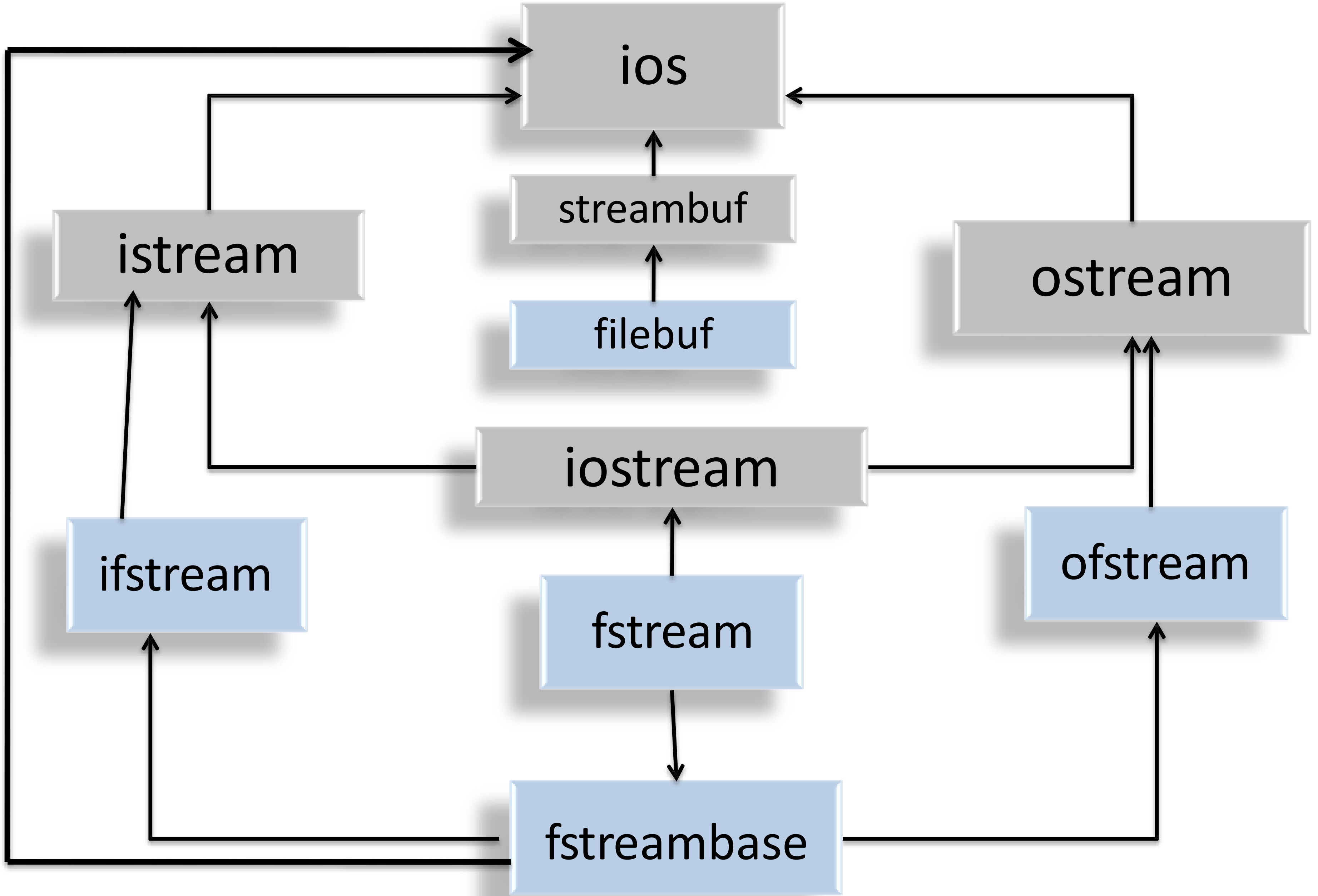
File I/O Operations



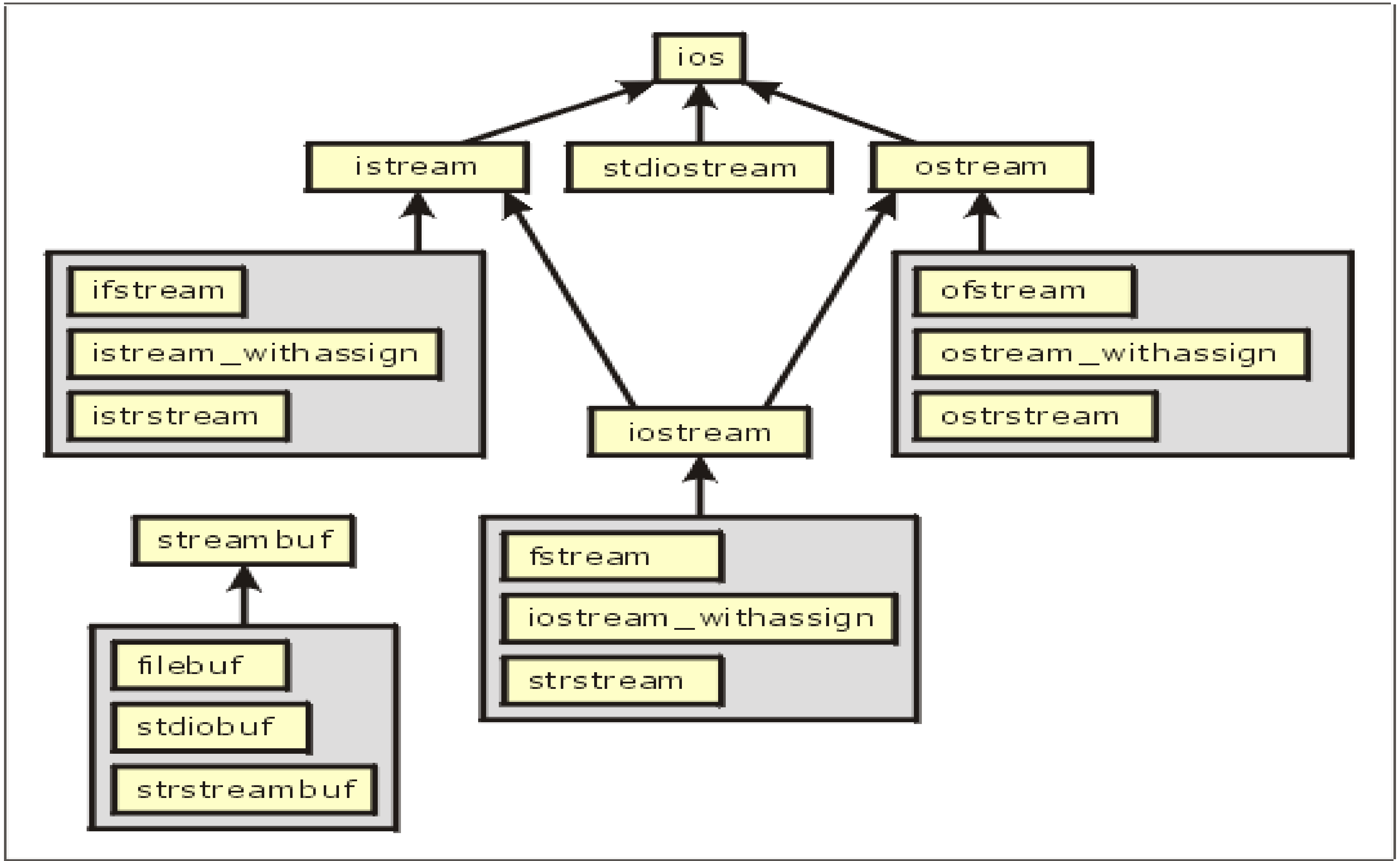
File Input & Output streams



I/O Stream classes for File operations



I/O Stream Class Hierarchy

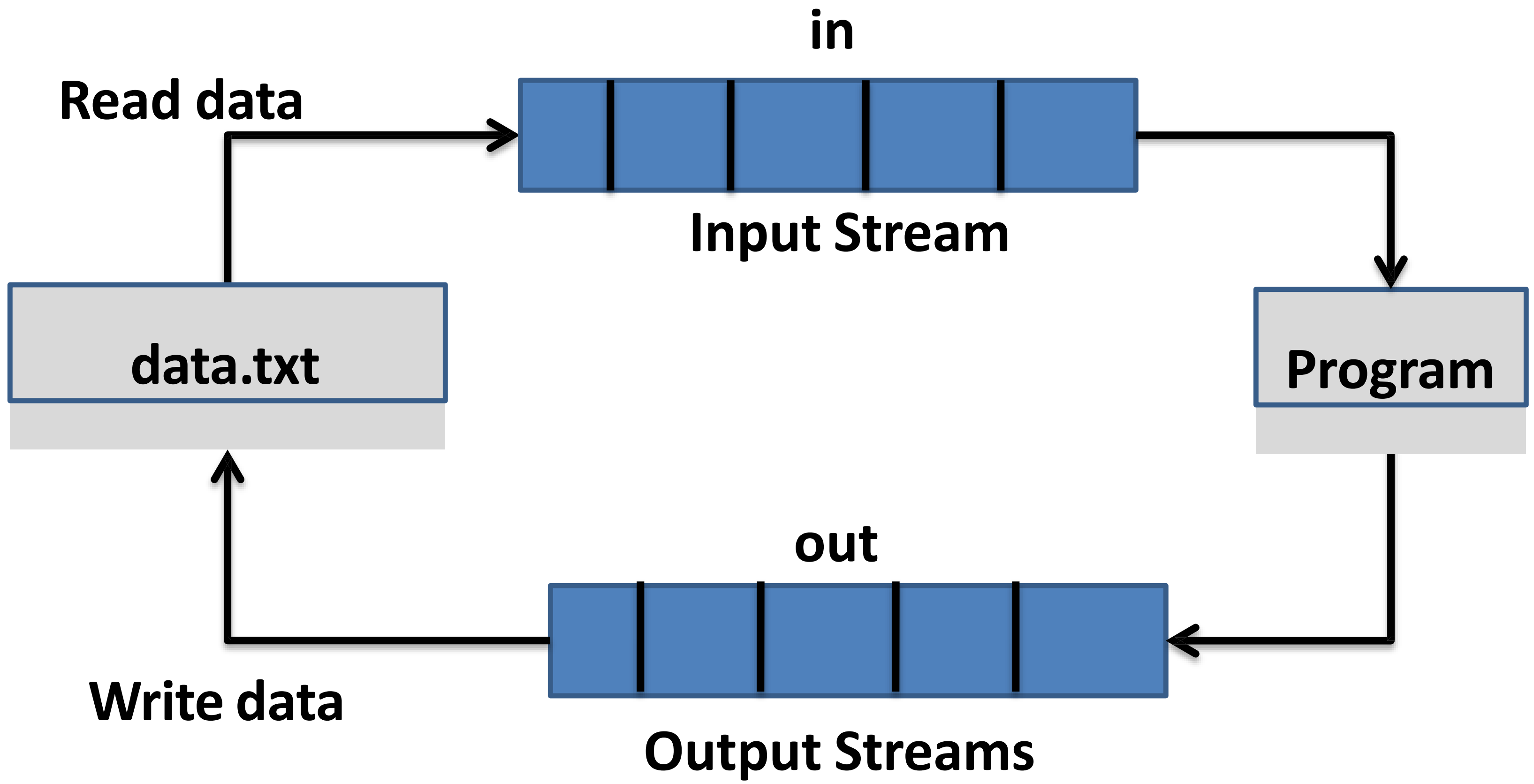


Opening & Closing a File

- **Opening (default mode):**
  - –Create a file stream
  - –Link it to the filename
  - –Two method to Open a file
    - Using constructor function of the class
    - Using member function **open()** of the class
- **Closing**
  - –Delinking the file stream from filename

Using constructor of the class

- ofstream out(“data.txt”);
- ifstream in(“data.txt”);



Using member function open() of the class

- **creating a filestream for writing**

```
ofstream  out;
out.open(“result.txt”,ios::app);
```

- **creating a filestream for reading**

```
ifstream in;
in.open(“inputdata.txt”,ios::app);
```

- **closing a file**
  - out.close( );
  - in.close( );

Modes of File Opening

Parameter	Meaning
ios::in	opens file for reading only
ios::out	opens file for writing only
ios::app	opens file for appending at the end only
ios::binary	opens file in binary mode
ios::trunc	Deletes the content of the file if it exists
ios::ate	opens file for appending but at anywhere

File Pointers

- Each file has two associated pointers
  - **get pointer** : to reads from file from given location
  - **put pointer** : to writes to file from given location
- **Manipulation of get pointer**
  - **seekg:** moves get pointer to a specified location
  - **tellg:** gives the current position of the get pointer
- **Manipulation of put pointer**
  - **seekp:** moves put pointer to a specified location
  - **tellp:** gives the current position of the put pointer

Moving to a specified location in file

- **Syntax:**
  - seekg(n_bytes);   //can be + or – n bytes
  - seekg(n_bytes, reposition);
- **repostion constants:**
  - **ios::beg**
  - **ios::cur**
  - **ios::end**

NOTE:
+ →go forward by n bytes
- → go backwards by n bytes

Error Handling with Files

- File which we are attempting to open for reading does not exist.
- The filename used for a new file may already exist.
- attempting an invalid operation such as reading past the eof.
- attempting to perform an operation when a file is not opened for that purpose.

Function	Return value & meaning
eof()	returns true (non-zero) if end-of-file encounterd while reading otherwise false(zero)
fail()	returns true when an iput of output operation has failed
bad()	returns true if an invalid operation is attempted of any unrecoverable error as occurred. if it false it may possible to recover from any other error reported and continue operation
good()	returns true if no error has occurred, if it false , no further operations can be carried out.