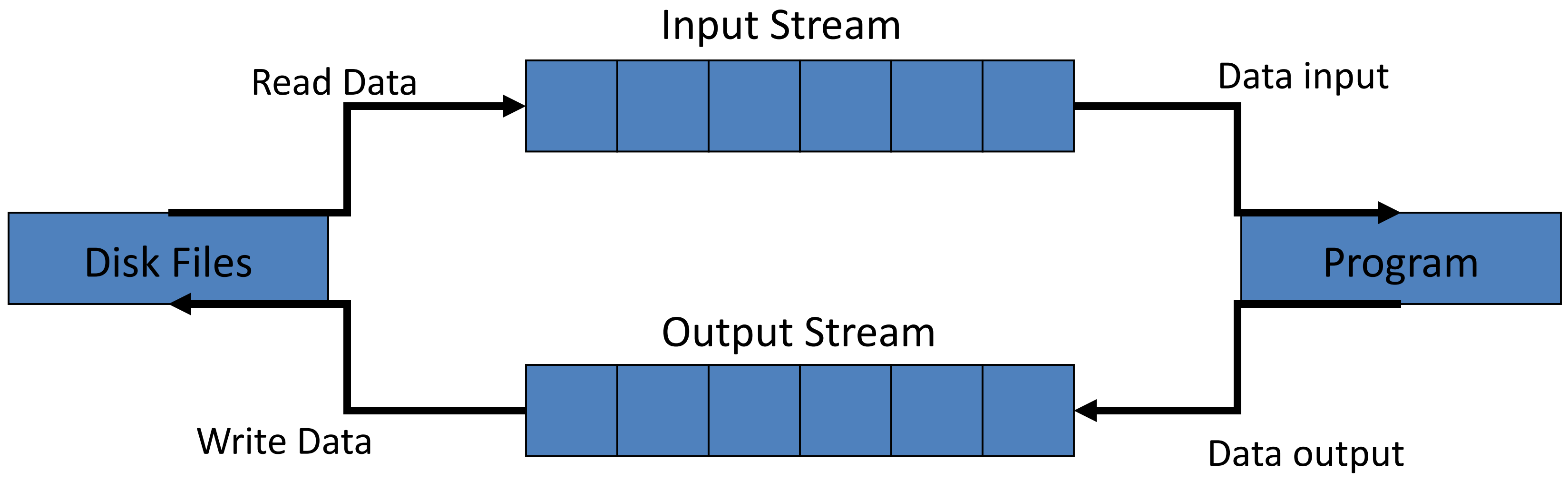
Learners Support Publications
www.lsp4you.com

Working with Fil

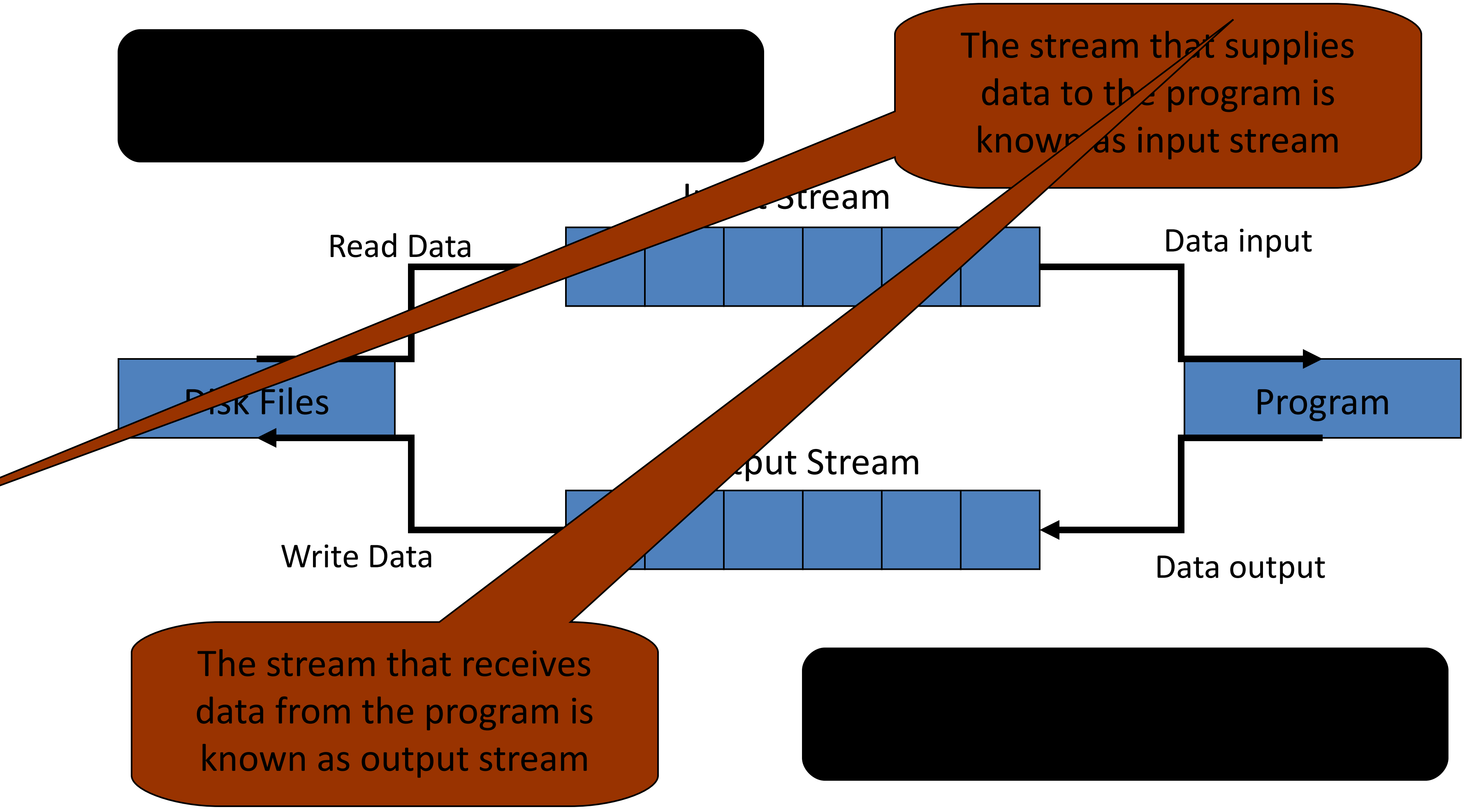
File Input and Output Stream

- C++ uses file streams as an interface between the programs and the data files.



File Input and Output Stream

[redacted]



The stream that supplies data to the program is known as input stream

The stream that receives data from the program is known as output stream

[redacted]

Opening Files

- For opening a file, we must first create a file stream and then link it to the filename.
- A file stream can be defined using the classes ifstream, ofstream, and fstream that are contained in the header file fstream.
- The class to be used depends on read or write.
- A file can be open in two ways:
  - Using the constructor function of the class.
    - Useful when we use only one file in the stream.
  - Using the member function open( ) of the class.
    - Use to manage multiple files using one stream.

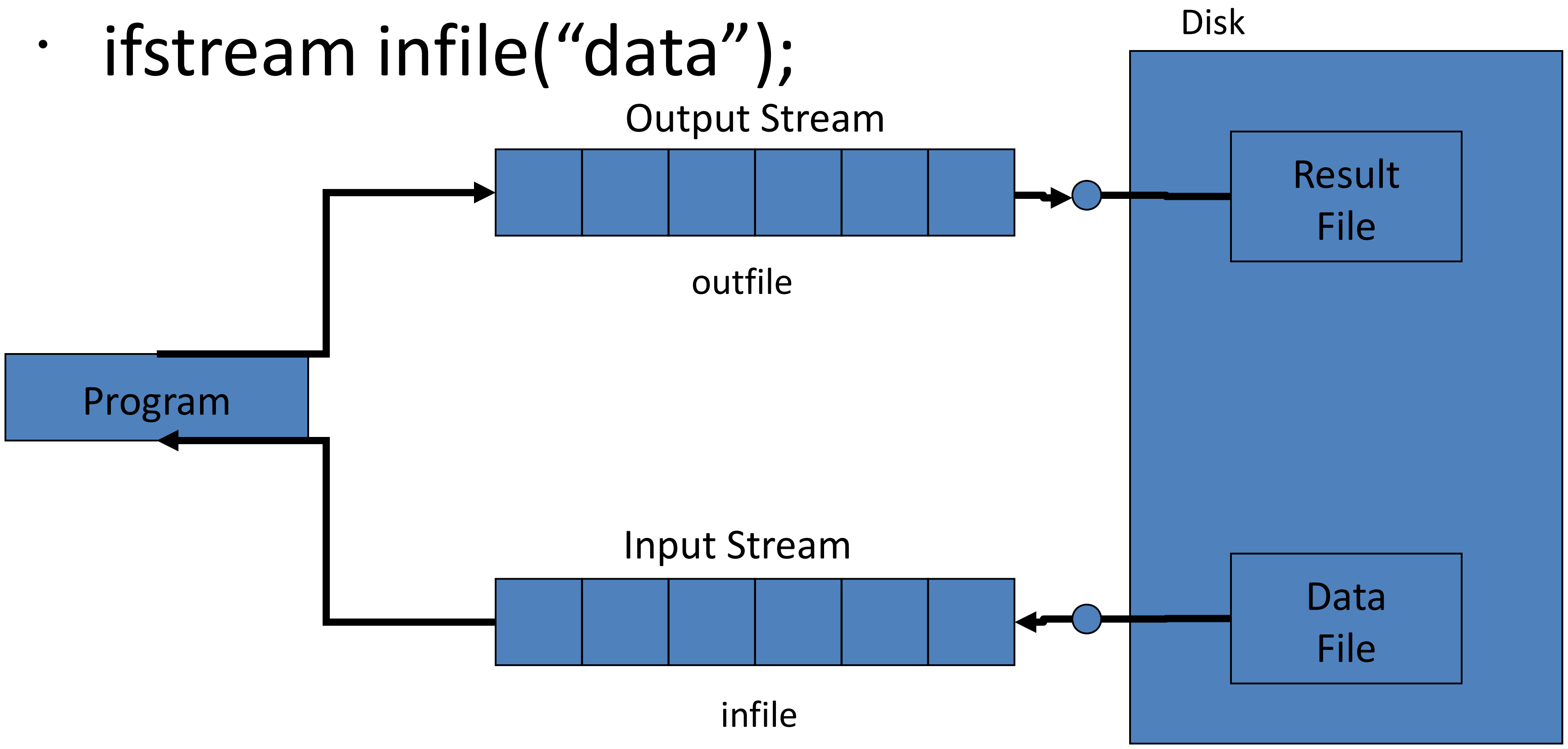
Opening Files Using Constructor

- This involves two steps:
  - Create a file stream object to manage the stream using appropriate class.
    - The class ofstream used to create output stream.
    - The class ifstream to create input stream.
  - Initialize the file object with the desired filename.

Opening Files Using Constructor

continue …

- ofstream outfile (“results”);
- ifstream infile(“data”);



Opening Files Using Open( )

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

- For processing a set files sequentially.

[redacted]

Opening Files Using Open( )

continue …

```
ofstream outfile;
outfile.open("DATA1");
……..
outfile.close( );
outfile.open("DATA2");
………
outfile.close( );
………
```

- The above program segment opens two files in sequence for writing the data.
- The first file is closed before opening the second one.

Detecting End-Of-File

- while (fin)
  - An ifstream object fin returns a value 0 if any error occurs in the file operation including the end-of-file condition.
  - So the while loop may terminates when fin returns a value of zero on reaching the end-of-file condition.

- if(fin1.eof() != 0) {exit(1);}
  - eof( ) is a member of ios class.
  - It returns a non-zero value if the end-of-file (EOF) condition is encountered, and a zero, otherwise.

File Modes

- stream-object.open("file_name", mode);
  - The second argument mode specifies the purpose for which the file is opened.
  - Default values for these second parameters:
    - ios::in – for ifstream - reading only
    - ios::out – for ofstream - writing only

File Modes

continue …

ios::app	→ **Append to end-of-file**
ios::ate	→ **Go to end-of-file on opening**
ios::binary	→ **Binary file**
ios::in	→ **Open file for reading only**
ios::nocreate	→ **Open fails if the file does not exist**
ios::noreplace	→ **Open files if the file already exists**
ios::out	→ **Open file for writing only**
ios::trunc	→ **Delete the contents of the file if it exists**

fout.open("data", ios::app | ios :: nocreate)

File Pointer

- Input Pointer (get pointer)
  - The input pointer is used for reading contents of a given file location.
- Output Pointer (put pointer)
  - The output pointer is used for writing to a given file location.

- Each time an input or output operation takes place, the appropriate pointer is automatically advanced.

File Pointer – Default Actions

- When a file opened in read-only mode, the input pointer is automatically set at the beginning of the file.
- When a file is opened in write-only mode, the existing contents are deleted and the output pointer is set at the beginning.
- When a file is opened in append mode, the output pointer moves to the end of file.

Functions for Manipulations of 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.

Seek Function with Absolute Position

- infile.seekg(10);

Moves the file pointer to the byte number 10. The bytes in a file are numbered beginning from zero. Therefore, the pointer pointing to the 11th byte in the file.

Seek Function with Specifying the Offset

- seekg(offset, refposition);
- seekp(offset, refposition);

The parameter offset represents the number of bytes the file pointer is to be moved from the location specified by the parameter refposition.

The refposition takes one of the following three constants defined in the ios class:

- ios : : beg → start of the file
- ios : : cur → current position of the pointer

Seek Function with Specifying the Offset

fout.seekg(0, ios : : beg);	Go to start
fout.seekg(0, ios : : cur);	Stay at the current position
fout.seekg(0, ios : : end);	Go to the end of file
fout.seekg(m, ios : : beg);	Move to (m+1)th 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

Sequential Input and Output Operations

- put( ) and get( ) Functions
  - The function put( ) writes a single character to the associated stream.
  - The function get( ) reads a single charracter from the associated stream.

Sequential Input and Output Operations

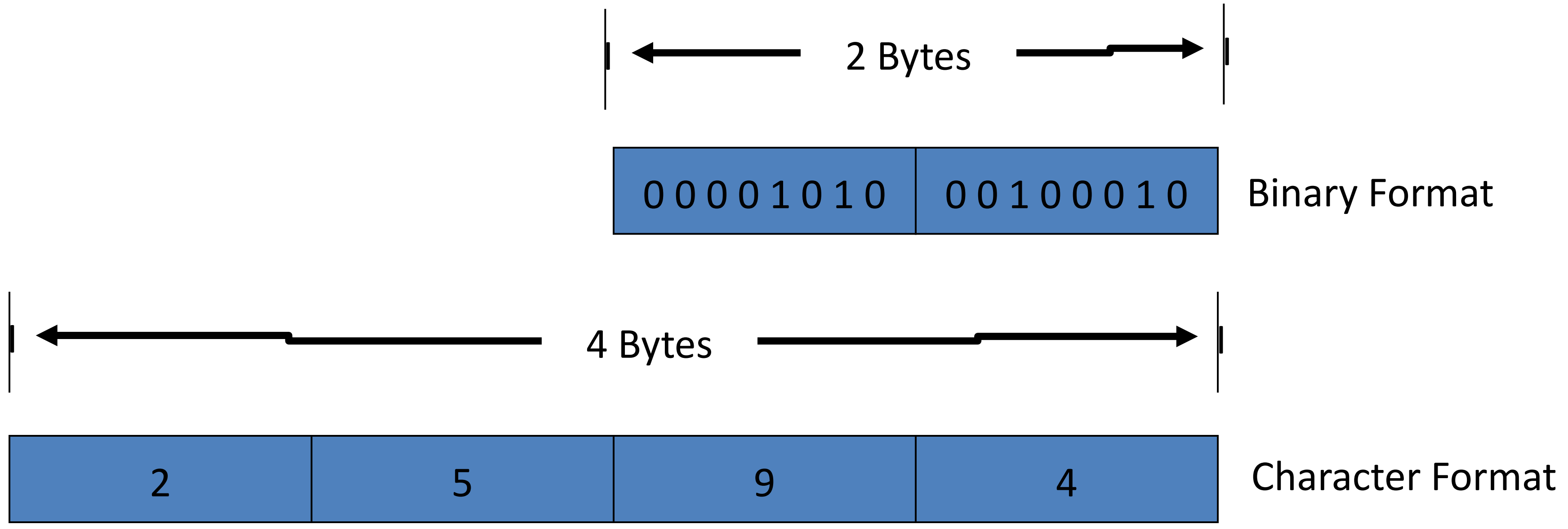
continue …

- write( ) and read( ) Functions
  - The functions write( ) and read ( ) handle the data in binary form.
  - The values are stored in the disk file in the same format in which they are stored in the internal memory.
  - An int takes two bytes to store its value in the binary form, irrespective of its size.
  - But a 4 digit int will take four bytes to store it in the character form.

Sequential Input and Output Operations

continue …

Representing 2594



Sequential Input and Output Operations

continue …

- infile.read((char *) &V, sizeof(V));

- outfile.write((char *) &V, sizeof(V));

- write( ) and read( ) functions take two arguments.
- First is the address of the variable V
- Second is the length of that variable in bytes.
- The address of the variable must be cast to type char * (pointer to character type).

Reading and Writing a Class Object

- The read( ) and write( ) are also used to read from or write to the disk files objects directly.
- The read( ) and write( ) handle the entire structure of an object as a single unit, using the computer’s internal representation of data.
- Only data members are written to the disk files.

Updating A File : Random Access

- The size of each object can be obtained using the statement

  int object_length = sizeof(object);

- The location of a desired object, say mth object

  int location = m * object_length;

  The location gives the byte number of the first byte of the mth object.

- Now we can use seekg( ) or seekp( ) to set the file pointer to reach this byte.

Updating A File : Random Access

continue …

- To find the total number of objects in a file using object_length

```
int n = file_size / object_length;
```

- The file size can be obtained using the function tellg( ) or tellp( ) when the pointer is located at the end of the file.

Error Handling During File Operations

- A file which we are attempting to open for reading does not exist.
- The file name used for a new file may already exist.
- We may attempt an invalid operation such as reading past the end-of-file.
- There may not be any space in the disk for storing more data.
- We may use an invalid file name.
- We may attempt to perform an operation when the file is not opened for that purpose.

Error Handling During File Operations

continue …

- The C++ file stream inherits a "stream_state" member from the class ios.
- This member records information on the status of a file that is being currently used.
- The class ios supports several member functions that can be used to read the status recorded in a file stream.
- eof( )
- fail( )
- bad( )
- good( ), etc.

Command – Line Arguments

- C++ supports a feature that facilitates the supply of argument to the main( ) function.
- These arguments are supplied at the time of invoking the program.
- They are typically used to pass the names of data files.
  - Eg:- exam data result
- The command-line arguments are typed by the user and are delimited by a space.

Command – Line Arguments

continue …

- The main function can take two arguments.
- main(int argc, char * argv [ ])
- The first argument argc (argument counter) represents the number of arguments in the command line.
- The second argument argv (argument vector) is an array of char type pointers that points to the command line arguments.

Command – Line Arguments

continue …

- C:\> exam data results
- The value of argc would be 3 and argv would be an array of three pointers to strings as:
  - argv[0] → exam
  - argv[1] → data
  - argv[2] → results

    - ……
    - ……
    - infile.open(argv[1]);
    - ……

Thank You