File I/O (Contd..)

Dr. Purushothama B R
Computer Science and Engineering
National Institute of Technology Goa

Ignore() function

- You can use the **ignore( )** member function to read and discard characters from the input stream.
- **Prototype:**
  - **istream &ignore(streamsize *num*=1, int_type *delim*=EOF);**

  - It reads and discards characters until either *num* characters have been ignored (1 by default) **OR**

  - The character specified by *delim* is encountered (**EOF** by default).
  - If the delimiting character is encountered, it is not removed from the input stream.
  - Here, **int_type** is defined as some form of integer.

peek( ) and putback( )

- You can obtain the next character in the input stream without removing it from that stream by using **peek( ).**
- **Prototype:**
  int_type peek( );
- It returns the next character in the stream or **EOF** if the end of the file is encountered.
- **int_type** is defined as some form of integer.
- You can return the last character read from a stream to that stream by using **putback( )**
- **Prototype:**

  istream &putback(char *c*);

c is the last character read.

flush( )

- When output is performed, data is not necessarily immediately written to the physical device linked to the stream.
- Instead, information is stored in an internal buffer until the buffer is full
- Only then are the contents of that buffer written to disk.
- However, you can force the information to be physically written to disk before the buffer is full by calling **flush( )**.
- Prototype:

  ostream &flush( );

.

Note

- Calls to **flush( )** might be warranted
  - when a program is going to be used in adverse environments.
  - for example, in situations where power outages occur frequently.

- *Closing a file or terminating a program also flushes all buffers.*

Random Access

- In C++'s I/O system, you perform random access by using the **seekg( )** and **seekp( )** functions.

- **Their most common forms are**

  istream &seekg(off_type *offset*, seekdir *origin*);
  ostream &seekp(off_type *offset*, seekdir *origin*);

- Here, **off_type** is an integer type defined by **ios** that is capable of containing the largest valid value that *offset* can have.

- **seekdir** is an enumeration defined by **ios** that determines how the seek will take place.

Random Access

- The C++ I/O system manages two pointers associated with a file.
- One is the *get pointer*, which specifies where in the file the next input operation will occur.
- The other is the *put pointer*, which specifies where in the file the next output operation will occur.
- Each time an input or output operation takes place, the appropriate pointer is automatically sequentially advanced.
- However, using the **seekg( )** and **seekp( )** functions allows you to access the file in a nonsequential fashion.

seekg()

- The seekg() function moves the associated file's current get pointer
  - offset number of characters from the specified origin, which must be one of these three values:
-

ios::beg	Beginning-of-file
ios::cur	Current location
ios::end	End-of-file

seekp()

- The seekp() function moves the associated file's current put pointer offset number of characters from the specified origin.
- Generally, random-access I/O should be performed only on those files opened for binary operations.
- The character translations that may occur on text files could cause a position request to be out of sync with the actual contents of the file.

Obtaining the Current File Position

- You can determine the current position of each file pointer by using these functions:
-
  pos_type tellg( );
  pos_type tellp( );

- Here, **pos_type** is a type defined by **ios** that is capable of holding the largest value that either function can return.
- You can use the values returned by **tellg( )** and **tellp( )** as arguments to the following forms of **seekg( )** and **seekp( )**, respectively