

C++ Input/Output & File I/O

ios_base class

typedefs

```
typedef T1 fmtflags;           T1, T2 are integer types (int, enum, ...)  
typedef T2 iostate;
```

constants

Format flag constants

These constants are used to assign a value to a `fmtflags` value. They represent formatting/parsing specifications for a stream.

<code>boolalpha</code>	reads or displays “true” or “false” for <code>bool</code> values instead of “1” or “0”.
<code>dec</code>	reads or displays integer input/output as a decimal number.
<code>fixed</code>	displays floating point numbers in decimal format.
<code>hex</code>	reads or displays integer input/output as a hexadecimal number.
<code>internal</code>	displays octal, hexadecimal, and scientific numbers in internal format.
<code>left</code>	left justifies output in a field.
<code>oct</code>	reads or displays integer input/output as an octal number.
<code>right</code>	right justifies output in a field.
<code>scientific</code>	displays floating point numbers in scientific (exponential) format.
<code>showbase</code> <code>for octal)</code>	displays a number base prefix for integer output (0x or 0X for hex and 0
<code>showpoint</code>	displays a decimal point in floating point numeric output.
<code>showpos</code>	displays a + sign for positive numeric output.
<code>skipws</code>	skips leading whitespace before >> operations.
<code>unitbuf</code>	flushes output stream after each insertion.
<code>uppercase</code>	displays hexadecimal output and the e of scientific format in uppercase.
<code>adjustfield</code>	group (bitwise or) of three flags : left, right, and internal.
<code>basefield</code>	group (bitwise or) of three flags : hex, oct, and dec.
<code>floatfield</code>	group (bitwise or) of two flags : fixed and scientific.

Stream state constants

These constants are used to assign a value to an `iostate` value. They represent the state of a stream.

`badbit` is set if a stream is corrupted.

eofbit is set if the EOF has been read.
failbit is set if an I/O operation fails.
goodbit is set if a stream is OK, no other bits set.

Some ios_base member functions

fmtflags flags() const;	returns the stream's fmtflags settings
fmtflags flags(fmtflags value);	sets the streams fmtflags. Note: clears any other flags already set.
fmtflags setf(fmtflags value);	sets the streams fmtflags. Note: does not clear other flags already set.
fmtflags setf(fmtflags val, fmtflags mask);	first clears the mask settings, then sets the stream's fmtflags. Note: does not clear other flags already set.
void unsetf(fmtflags value);	clears the fmtflags values(s).
streamsize precision() const;	returns the stream's precision setting for floating point values. Note: precision setting is the number of decimal places if fixed or scientific is set. If neither is set, the precision represents the number of significant digits. In either case, the value is automatically rounded to the precision setting.
streamsize precision(streamsize size);	sets the stream's precision. See note above.
streamsize width() const;	returns the stream's field width that applies to the next output value.
streamsize width(streamsize size);	sets the stream's <i>minimum</i> field width that applies only to the next output value. The width applies to both numeric and char data.

Some basic_ios member functions

The **ios** class is a typedef for the instantiation of the basic_ios template of type char.

<code>char_type fill() const;</code>	returns the fill character assigned to a stream. This character is used to pad a field whose width is more than the number of characters needed to display a value.
<code>char_type fill(char_type ch);</code>	sets the fill character assigned to a stream.
<code>bool operator!() const;</code>	returns <code>fail()</code> .
<code>bool bad() const;</code>	returns true if the stream's badbit is set. The <code>bad()</code> function indicates a corrupted stream.
<code>void clear(iostream state=goodbit);</code>	sets the iostate (by default to goodbit).
<code>bool eof() const;</code>	returns true if the eofbit is set (the stream's EOF has been read).
<code>bool fail() const;</code>	returns true if the stream's failbit or badbit is set. Use the <code>fail()</code> function to check to see if a file is successfully opened.
<code>bool good() const;</code>	returns true if the stream's goodbit is set.
<code>iosstate rdstate() const;</code>	returns a stream's iostate value.

Some basic_istream member functions

The **istream** class is a typedef for the instantiation of the `basic_istream` template of type `char`.

<code>streamsize gcount() const;</code>	returns the number of characters read by the last input operation
<code>int get();</code>	reads and returns the next character available in a stream. Returns EOF if no character is available.

istream specific functions

<code>istream& get(char& ch);</code>	reads the next char and assigns it to <code>ch</code> . Returns the "current" <code>istream</code> .
--	--

`istream& get(char* buf, streamsize n);` reads n-1 bytes and stores them in buf. A newline(\n) will terminate the read. In that case, the newline **is not read and not stored**. The char data in buf is null-terminated.

`istream& get(char* buf, streamsize n, char delimiter);` reads at most n-1 bytes and stores them in buf. If the delimiter is encountered, the read ends. The delimiter **is not read and not stored**. The char data in buf is null-terminated.

`istream& getline(char* buf, streamsize n);` reads n-1 bytes and stores them in buf. A newline(\n) will terminate the read. In that case, the newline **is read, but not stored**. The char data in buf is null-terminated. **The newline (\n) is considered the delimiter for this function.** See warning in next paragraph.

`istream& getline(char* buf, streamsize n, char delimiter);` reads at most n-1 bytes and stores them in buf. If the delimiter is encountered, the read ends. The delimiter **is read and not stored**. The char data in buf is null-terminated.

Warning: If the delimiter is not read, it is considered an error. The failbit is set, even though the stream data is store in buf. This warning does not apply to the get() function.

`istream& ignore(streamsize n=1, int_type delimiter=EOF);` extracts and discards n char or extracts until delimiter is read.

`int_type peek();` returns the next available char. Does not increment the current get pointer.

`istream& putback(char ch);` inserts the char ch into the input stream at the current get pointer position-1. Moves the current get pointer position back 1 byte.

istream& read(char* buf, streamsize n); reads n bytes into buf. buf is not null terminated.

streamsize readsome(char* buf, streamsize n); reads n bytes into buf. buf is not null terminated. Returns the number of characters read. The difference between read() and readsome() is that readsome() does not set the failbit if it cannot read n characters (if it encounters EOF before the read is complete).

istream& unget(); inserts the last char read into the input stream at its original position.

Some basic_ostream member functions

The **ostream** class is a typedef for the instantiation of the basic_ostream template of type char.

ostream& flush(); writes any data in the output buffer to the output stream.

ostream specific functions

ostream& put(char ch); writes ch to the output stream.

ostream& write(char* buf, streamsize n); writes n bytes of char starting from the address buf into the output stream.

Input/Output Manipulators

Manipulators are functions or function-like operators that change the state of the I/O stream. Those manipulators with arguments require the <iomanip> header file.

Manipulator	I/O	Purpose
boolalpha	I/O	sets boolalpha flag
dec	I/O	sets dec flag for i/o of integers, clears oct,hex
endl	O	inserts a newline and flushes output stream
ends	O	inserts a null
fixed	O	sets fixed flag
flush	O	flushes stream

hex	I/O	sets hex flag for i/o of integers, clears dec,oct
internal	O	sets internal flag
left	O	sets left flag
noboolalpha	I/O	clears boolalpha flag
noshowbase	O	clears showbase flag
noshowpoint	O	clears showpoint flag
noshowpos	O	clears showpos flag
noskipws	I	clears skipws flag
nounitbuf	O	clears unitbuf flag
nouppercase	O	clears uppercase flag
oct	I/O	sets oct flag for i/o of integers, clears dec,hex
resetiosflags(ios_base::fmtflags mask)	I/O	clears format flags specified by mask
right	O	sets right flag
scientific	O	sets scientific flag
setbase(int base)	I/O	sets integer base (8, 10, or 16)
setfill(char_type ch)	O	sets the fill character to ch
setiosflags(ios::base::fmtflags mask)	I/O	sets format flags to mask value
setprecision(int p)	O	sets precision of floating point numbers
setw(int w)	O	sets output field width to w
showbase	O	sets showbase flag
showpoint	O	sets showpoint flag
showpos	O	sets showpos flag
skipws	I	sets skipws flag
unitbuf	O	sets unitbuf flag
uppercase	O	sets uppercase flag
ws	I	extracts whitespace

Overloading the Insertion and Extraction Operators

C++ File I/O

basic_ifstream<> members

basic_ifstream();

explicit basic_ifstream(const char* filename, ios_base::openmode mode = ios_base::in);¹

¹ explicit means that an object of this type can only be created by an explicit declaration of the object, and not by a conversion

```
void close();
```

```
bool is_open();
```

```
void open(const char* filename, ios_base::openmode mode = ios_base::in);
```

basic_ofstream<> members

```
basic_ofstream();
```

```
explicit basic_ofstream(const char* filename, ios_base::openmode mode =  
ios_base::out);
```

```
void close();
```

```
bool is_open();
```

```
void open(const char* filename, ios_base::openmode mode = ios_base::out);
```

basic_fstream<> members

```
basic_fstream();
```

```
explicit
```

```
basic_fstream(const char* filename, ios_base::openmode mode = ios_base::in |  
ios_base::out);
```

```
void close();
```

```
bool is_open();
```

```
void open(const char* filename, ios_base::openmode mode = ios_base::in |  
ios_base::out);
```

More I/O Members and Types

ios_base class

typedefs

typedef T3 openmode;

constants

Open mode constants

These constants are used to assign a value to an openmode value. They represent the mode for opening a stream.

app	position to the end of the stream before each write operation.
ate	position to the end of the stream when the stream is opened.
binary	open the stream in binary mode (newlines are 1 byte).
in	open for input.
out	open for output
trunc	delete an existing file when opening.

Positioning constants

These constants are used to assign a value to a seekdir value. They used for relative positioning in a file stream with the seekg() and seekp() functions.

beg	position is relative to the beginning of a file stream.
cur	position is relative to the current position in a file stream.
end	position is relative to the end of a file stream.

More basic_istream members

istream& seekg(ios_base::pos_type pos); positions to the location indicated by pos in a file stream. pos_type is the type returned by the tellg() function.

istream& seekg(ios_base::pos_type pos, ios_base::seekdir dir);
seeks to the position pos characters from dir in a file stream.

pos_type tellg(); returns the character position in the file stream. If the stream state is non-zero, then the function returns pos_type (-1).

More basic_ostream members

`ostream& seekp(ios_base::pos_type pos);` positions to the location indicated by `pos` in a file stream. `pos_type` is the type returned by the `tellp()` function.

`ostream& seekp(ios_base::pos_type pos, ios_base::seekdir dir);` seeks to the position `pos` characters from `dir` in a file stream.

`pos_type tellp();` returns the character position in the file stream. If the stream state is non-zero, then the function returns `pos_type (-1)`.