A Summary of Standard I/O for C++

classes

`ostream`, `istream`, `iostream` are declared in the header file `iostream.h`.

They inherit common functions fom the class `ios`.

objects

`cout`, `cerr`, `clog` are instances of `ostream`.

`cin` is an instance of `istream`.

insertion into an output stream

```
cout << n;
```

`n` can be any base type, or a pointer to char (for string output)

insertions can be chained:

```
cout << n << p << q;
```

format flags

`cout` and `cin` contain several single-bit flags for format control. They are named in an enum in class `ios` as:

```
ios::skipws     // skips whitspace on input
ios::left       // left justification
ios::right      // right justifiction
ios::internal   // pads after sign or base character
ios::dec        // decimal format for integers
ios::oct        // octal format for integers
ios::hex        // hex format for integers
ios::showbase   // show the base character for octal or hex
ios::showpoint  // show the decimal point for all floats
ios::uppercase  // uppercase A-F for hex
ios::showpos    // show +ve sign for numbers
ios::scientific // use exponential notation
ios::fixed      // used oridnary decimal notation
ios::unitbuf    // flush the buffer
```

There are also three combination fields:

```
ios::basefield    = ios::dec | ios::oct | ios::hex
ios::adjustfield  = ios::left | ios::right | ios::internal
ios::floatfield   = ios::scientific | ios::fixed
```

These flags can be set with the member function setf. e.g.

```
cout.setf(ios::showpos);
```

They can be or'd together:

```
cout.setf(ios::showpos | ios::uppercase);
```

or some bits can be unset while one is being set:

```
cout.setf(ios::oct, ios::dec | ios::oct | ios::hex);
```

This sets the bit for `oct`, after unsetting the bits for `oct`, `dec` and `hex`, ensuring that only one of the bits is turned on.

`unsetf` turns bits off:

cout.unsetf(ios::showpos);

field width

```
cout.width(20);
```

or, using a manipulator:

```
cout << setw(20);
```

These only change the width for the next item output only.

fill character

```
cout.fill('0');
```

or

```
cout << setfill('0');
```

justification

```
cout.setf(ios::right, ios::adjustfield);
```

precision

The format of floats can be set by:

```
cout.setf(ios::scientific, ios::floatfield);
```

and the number of decimal places by:

```
cout.precision(2);
```

or:

```
cout << setprecision(2);
```

extraction from an input stream

```
cin >> n;
```

`n` can be any base type, or a pointer to char (for string input). Whitespace characers (blank, newline, tab) are ignored if they precede input, and the first whitespace charcter after the input is used a stopper - it is not read, but is left for the next attempted input. Extractions can be chained:

```
cin >> n >> p >> q;
```

Also the function get can be used to read any character, including whitespace

```
cin.get(c);
```

extraction of strings

Strings can be input teminated by whitespace, or limited by using width. e.g.

```
char s[21];
cin.width(20);
cin >> s;
```

Also, get can be used:

```
char s[21];
cin.get(s, 20);
```

end of file

When extraction fails to read a value from the input stream, the eof flag in cin is set. A typical way to handle this is:

```
int n;
cin >> n;
while (!cin.eof())
  cin >> n;
```

errors on input

If an unexpected character is encountered or end-of-file occurs, then the stream is in a failure state and can be tested by:

```
if (cin.fail()) ...
```

Another way to do the same thing is to use the stream itself, converted to an integer:

```
if (!cin) ...
```

This is the same as using the `fail` function.