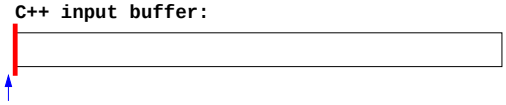
Formatted input example (4 pages)

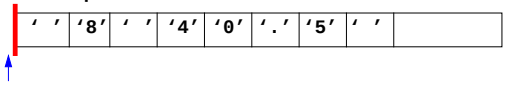
```
// our program
string name;
float weight;
int age;
cin >> age >> weight >> name;
```

C++ input buffer:



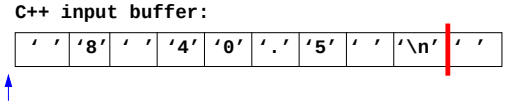
The above picture represent the situation at the start of the program. The blue arrow is the position where `cin` will start reading next time. Characters behind the bar are not available to `cin` yet, `cin` will stop and wait when reaching the bar. Now the user enters `" 8 40.5 "`:

C++ input buffer:



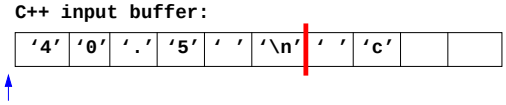
Nothing happen since `cin` is still blocked by the red bar. Now the user press Enter, followed by a space `" "`.

C++ input buffer:



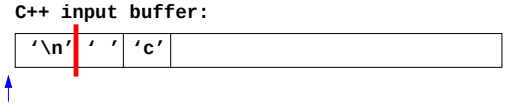
The buffer picture after the leading blanks is removed:

C++ input buffer:



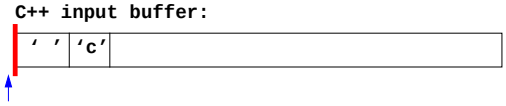
A floating point value can contain digits and a single dot. All matching characters are read and interpreted. *40.5* is stored in *weight*.

C++ input buffer:



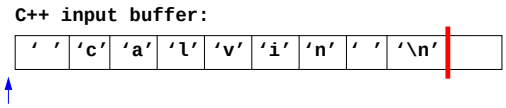
Characters are still available. The next thing to read is a string. When formatted input (`>>`) is used all leading blank characters will be skipped. It will then read the next word. A word is every character until the next blank character. In this case the newline character is read. But then no more characters are available.

C++ input buffer:



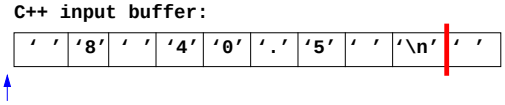
The user enters `"alvin "` and press enter.

C++ input buffer:



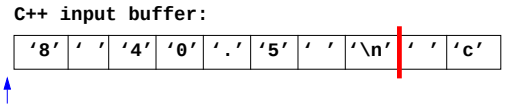
(Buffer from last page repeated here for convenience.)

C++ input buffer:



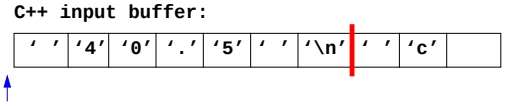
The blocking bar is moved to just after the newline character. `cin` will now start reading from the buffer. The characters read are removed from the buffer. The first thing to read is an integer (*age*). Since formatted input (`>>`) is used all blank characters before the integer will be removed first. Meanwhile the user enters `"c"`:

C++ input buffer:



It is now time to interpret the next characters as integers. the 8 will be read, but as the following space is not valid in a integer the reading will stop. *8* is stored in *age*.

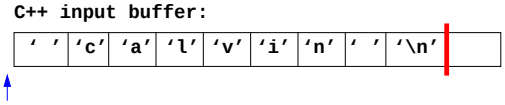
C++ input buffer:



There is still unblocked characters available in the buffer, `cin` will continue to read the next value (*weight*), that should be interpreted as float. Since formatted input (`>>`) is used all leading blank characters is removed first.

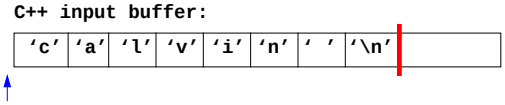
(Buffer from last page repeated here for convenience.)

C++ input buffer:



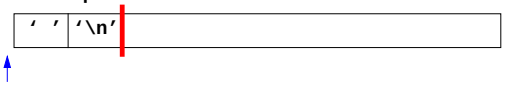
`cin` will continue removing leading blanks.

C++ input buffer:



Then `"calvin"` is read stored to the *name* variable. `cin` will stop at the first blank character.

C++ input buffer:



Now `cin` is done with all requested input. The characters still in the buffer will remain and be read when `cin` is used again.
N.B! The remaining characters may come as a surprise to you the next time you try to read something.