Q) How streams work?

A) Streams are serial interfaces to storage, buffers files, or any other storage medium. The difference between storage media is intentionally hidden by the interface; you may not even know what kind of storage you're working with but the interface is exactly the same.

IO streams not only define the relation between a stream of characters and the standard data types but also allows you to define a relationship between a stream of characters and your own classes.

The "serial" nature of streams is a very important element of their interface. You cannot directly make random access random reads or writes in a stream (unlike, say, using an array index to access any value you want) although you can seek to a position in a stream and perform a read at that point.

Using a serial representation gives a consistent interface for all devices. Many devices have the capability of both producing and consuming data at the same time; if data is being continually produced, the simplest way to think about reading that data is by doing a fetch of the next characters in a stream. If that data hasn't been produced yet (the user hasn't typed anything, or the network is still busy processing a packet), you wait for more data to become available, and the read will return that data. Even if you try to seek past the end (or beginning) of a stream, the stream pointer (i.e. get or put pointer) will remain at the boundary, making the situation safe. (Compare this with accessing data off the end of an array, where the behavior is undefined.)

*Refer stream docs*

----------------------------------------------------------------------------------------------------------------------------------------

Q) initializing static variable with a function call gives compilation error?

#include <stdio.h>

int foo(){

return 1;

}

int main(void) {

static int q = foo();

return 0;

}

This is a C code and not C++. It compiles and run fine in C++ but not C.

A) Global and static variables can only be initialized with constant expressions known at compile time. Calling your foo() function does not constitute using a constant expression. Further, the order in which global and static variables are initialized is not specified. Generally, calling foo() would mean that there must be a certain order, because the function can reasonably expect some other variables to be already initialized.

In Other Words, in C, neither of your code is executed before main().

In C++ there are ways around it, but not in C.

static int (\*q) (void);q=foo;

==============================================================================