The output of c++ program that uses fstream to read file:

```
Count of char of 'a' is 19082160

Time taken: 46.48s
```

The output of c program that uses fopen to read file:

```
Count of char of 'a' is 19082160

Time taken: 13.91s
```

The output of c program that uses memory mapping to read file:

```
Count of char of 'a' is 19082160

Time taken: 1.49s
```

In the first method (C++ that uses fstream) the advantage is easiness of implementation (it is the first thing that someone learns when beginning to code), it is more adoptable and can be used in Windows, as well. However, as can be observed by the times that they are lasting that method is far slower than the other two methods.

In the second method (C that uses fopen) the advantage is again implementation easiness, although it takes little bit more time to understand it compared to C++ (for 2 terms, C++ is being familiarized, but C is similar to that language of course). In addition to that, only one library is enough to compile that program (stdio.h) and no part of RAM is used to map. Also, it utilizes the file more than C++, that can be seen from the times again, that method is 3 times faster than C++ one. However, it is still slow compared to memory mapping.

Finally, last method (memory mapping in C) is far more efficient than others (10 times faster than second and 30 times faster than first one). It makes a part of RAM, buffer available so that file can be mapped there to fasten the I/O operations by enabling the access from the RAM not from the disk. However, it uses memory from the RAM which makes its space complexity larger than the others.

The results are obviously showing that fstream is 3 times slower than fopen. This may be caused from the typenames. Fstream uses ifstream as a type whereas, FILE * is used by c language. It can be said that FILE * type is strictly efficient than ifstream type.

On the other hand, memory mapping fastens the process 10 times even than the fopen method. The reason behind is simple: It copies file content to the buffer in order to make that file more accessible. Therefore, while doing I/O operations in that specific file, the access is 10 times faster since you are using the buffer. Process is like a portion of virtual memory is assigned directly to portion of a file and OS can reference that portion as it is a part of main memory (because of the buffer). After that mapping, file can be used in program like an array that are actually in main memory thanks to the process address space and its corresponding file.

Furthermore, file size has a great impact. 266 MB txt file requires a lot of I/O operations even with memory mapping it takes more than one second time. If the file is small (like 2MB) the difference between C++ and C would become ignorable (C will take 1.5 secs, where C++ will take 4.5 sec, 3 sec is nothing compared to 30 secs).

This shows that, when working with larger files is needed, memory mapping is a must, but dealing with smaller files is easy, so implementation of memory mapping and using extra memory may not be desired that much.