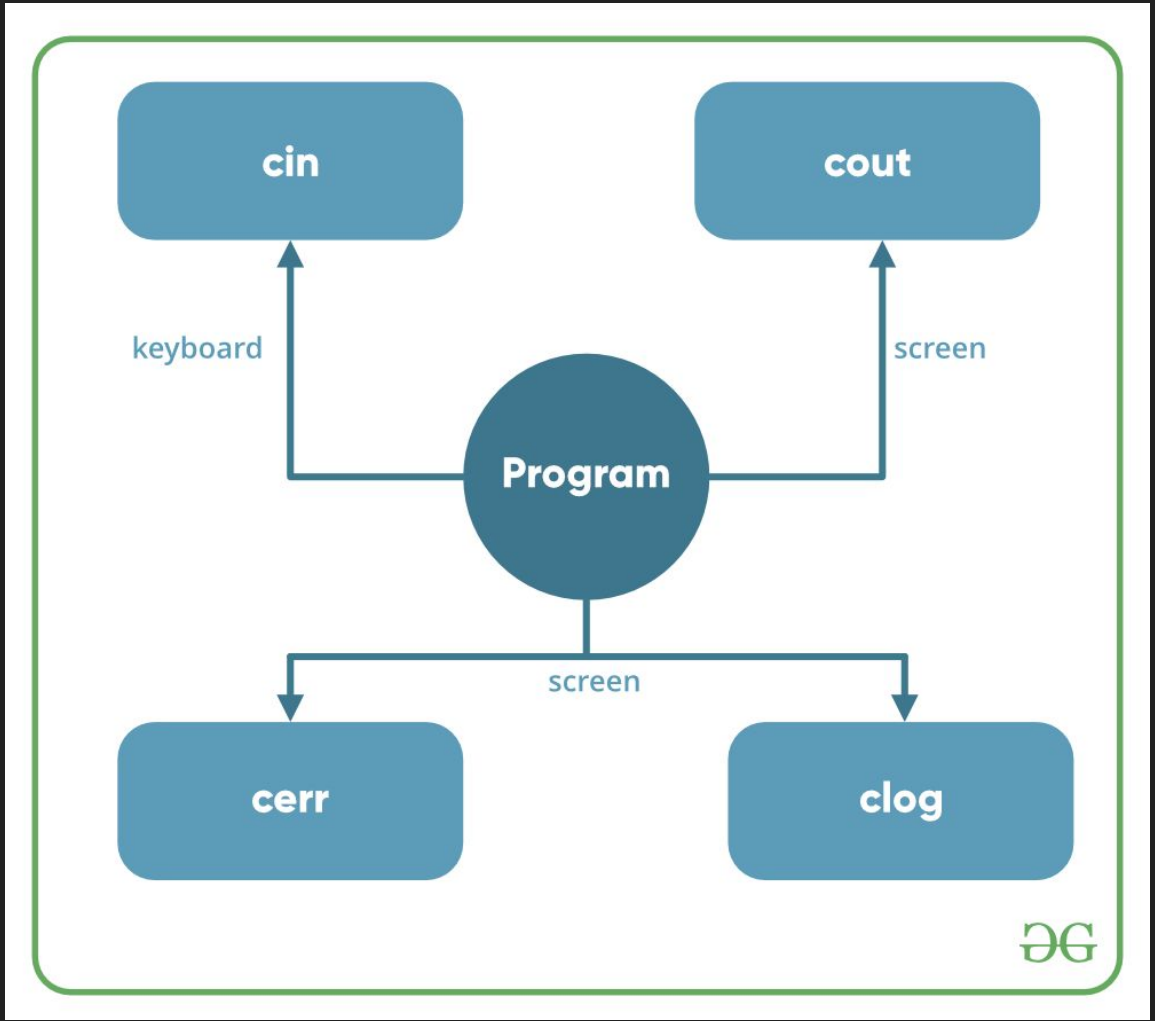
Object Oriented Programming

C++ files and streams

Input / Output streams

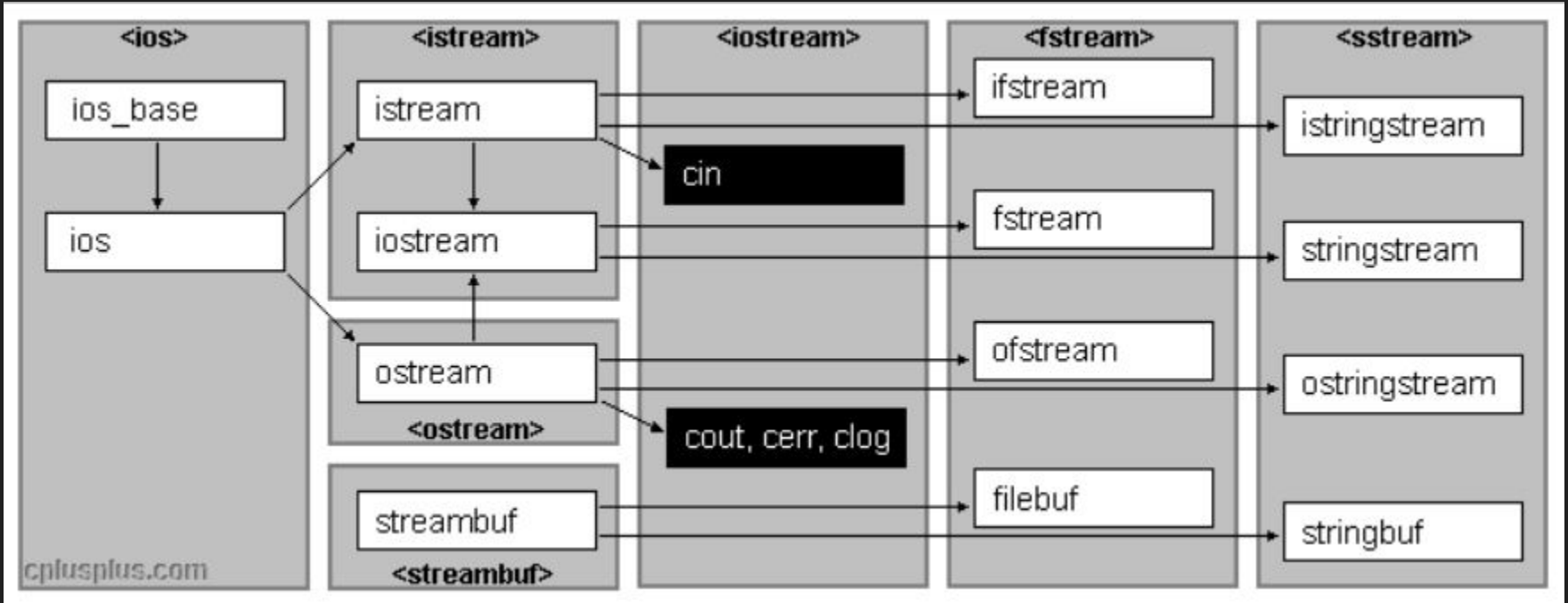
The truth about cin / cout


cin
cout
keyboard
screen
Program
screen
cerr
clog

iostream standard library overview

- So far we have used the console to interact with the user in two ways:
  - input some values from the keyboard
  - output results from computation on the console
- As we have noticed, *std::cin* and *std::cout* require *iostream* header to operate with them as mechanisms for input / output operations
- *iostream* - header hat defines the standard *input / output* **objects**
  - ***std::cin***
  - ***std::cout***
- The iostream is an object oriented library that provides input and output functionality using **streams**.

Stream classes hierarchy



File streams

I’m old enough to be RAM independent...

fstream standard library overview

- *fstream* is a library that defines three new **data types (classes)** but this time for interacting with **file streams**
- ***std::ofstream*** class represents the **output** file stream and is used to **create** files and to **write** information to files.
- ***std::ifstream*** class represents the **input** file stream and is used to **read** information from files.
- ***std::fstream*** class combines both the previous classes and is capable of creating files, writing information to them, and reading information from them.
- To perform file processing in C++, header files <iostream> and <fstream> must be included in your C++ source file.

Operations with a file

- Opening a file by file path
  - passing the path to the file to the parameter constructor of the file stream class
  - passing the path to the file to *open()* method of the file stream class
- Opening a file with a specific **mode:**
  - *std::ios::in* - opens file for **reading**
  - *std::ios::out* - opens the file for **writing**
  - *std::ios::binary* - opens the file in binary mode, rather than text file
  - *std::ios::ate* - opens the file and sets the initial position at the end of the file
  - *std::ios::app* - all output operations are performed at the end of the file, appending the content to the current content of the file.
  - *std::ios::trunc* - if the file exists, its content gets deleted
  - modes can be combined with the “|” operator

Operations with a file contd.

- Closing an opened file
  - with the standard member function *close()*
- Writing to an opened file
  - like the writing to the console, with the *stream insertion operator (<<),* but this time used over **ofstream** or **fstream** objects
- Reading from an opened file
  - like the reading from the keyboard, with the *stream extraction operator (>>),* but this time used over **ifstream** or **fstream** objects
- After all the work done with the files, they must be closed for security purposes :)

Flags for checking file status

- is_open() - checks whether the file is open, or not
- bad() - checks whether the input / output operation has failed
- fail() - like bad, but more strict - checks for type matches
- eof() - returns true if a file open for reading has reached the end
- good() - it returns false if any of the flags *bad(), fail() or eof()* returns true

Input / output streams for User defined types

- *Input stream - operator >> overloading*

  friend std::istream& operator >> (std::istream&, Vector<T>&)

- *Output stream - operator << overloading*

  friend std::ostream& operator << (std::ostream&, const Vector<T>&)

- Why friend functions?
- Why returning reference to i/ostream classes?
- Why constness of ofstream and user defined classes?