C++ Cheat sheet:

1. C++ Character Set:

A character set is a set of valid symbols that make up the basic components of the language that we are about to learn. Each language can have a different set of symbols according to their specific needs but are more likely the same except for a few exceptions.

The C++ Character set includes Letters(A-Z,a-z) , Digits/numbers(0-9), Special symbols and White spaces.

2. C++ Tokens:

As we now understand about the character set, we divide the character set into specific categories depending upon their functionality/working. These categories are known as Tokens and are the smallest component or individual unit that exists in the c++ program to be written.
The various types of tokens are:

a. Keywords: Think of them as reserved words, that is, words you cannot use however you wish and can be used to perform specific tasks only. Some examples can be: if, else, for, char, double etc.

b. Identifiers: These are user defined names given to some entities such as variables, functions etc. They can be anything to your liking and coding preference but some rules need to be followed:
   i. First character must be a letter or an underscore(_)
   ii. They cannot be same as keywords
   iii. Cannot have space between them or special characters except for the underscore
   iv. They are case Sensitive (Very very important)

c. Constants: These are identifiers that represent a constant value which means the value stored inside these variables cannot be changed during the execution of the program. A good mathematical example is the value of Pi which is 3.14 and does not change. We define a Constant in C++ using the const keyword.

d. String: These are variables that store textual data and are used to manipulate them. “Text” is a sequence of characters. We define a string variable in C++ using the “string” Keyword.

e. Operators: These are symbols that tell the machine what to do with the data that you have provided.

   They can be of various kinds:
   i. Arithmetic operators: These symbols are used to perform arithmetic operations. Some common examples are: +,-,*,/,%
   ii. Assignment operators: These operators are used to assign values to variables. Some assignment operators can be used to perform arithmetic operations along with assigning values to variables. Some examples are: =, +=, -=, *=, /= , %=.
   iii. Relational Operators: These operators are used to compare given values and return either “TRUE” or “FALSE” as a result. Some examples are: ==, <, >, !=. <= , >=
   iv. Logical Operators: These are conditional operators which are used to check whether the given conditions are satisfied or not. We mostly use them in conditional statements. For example: &&, ||, ! . They also return their result as either “TRUE” or “FALSE”.

   v. Bitwise operators: These operators are used to perform operations on individual bits. For example &, |, ^, ~, << , >>

3. Structure of a C++ Program:

The structure of a C++ program is very similar to any other programming language. Some parts of the program and how they are written will be typically based on your preference while others are rigid and have to follow certain predefined standards.

The Beginning of any program includes the internal documentation which defines the functioning of the program and logic used to build the program. It is written in comments so the documentation part is not compiled and can only be read. This part remains in personal choice yet in the perspective of examination I would recommend using a bring intro documentation above the written code.

Include files: Include files or header files provide the user with many in built functionalities that are defined in the c++ library. For using these header files we must include the file into our program with the syntax #include<'Header_file_name'>.

NOTE: Namespaces are also part of include files that help to group certain objects and define their scope in the program.

There are various header files that can be included in the program but including unnecessary header files into the program that won't be needed would slow down the execution of the program.

Some of the header files are:

a. <iostream> : This header is used to declare objects that control reading from and writing to the standard streams. This is the basic header that is needed to perform input and output operations in the program. Some objects under the iostream header are: **cin, cout,** cerr, clog, wcin, wcout etc.
Stream: It is a sequence of characters read from or written to an IO device.

   cin: used to take the input stream. We use ">>" with the cin object which is known as the 'input operator'. We can also use multiple input operators to take multiple inputs by pressing space or enter.

   cout : used to output a stream. We use and output operator "<<" along with it to print something on the screen. Remember that while printing statements that contain various types of data we separate the various types using the output operator.

b. <iomanip>: This is a header file used to manipulate the output of C++ program. Though manipulators like endl are defined in the iostream header file some other manipulators such as setw() are defined in the iomanip header file.

   The setw() inserts whitespaces between two variables. It takes arguments as integer value and performs the specific task.

4. Main function: main():

The main function is the first function that is executed by the compiler. The body of the main function or any other function is defined using curly braces. The main function is similar to other functions just with the highest priority.

Functions: These are block of codes that perform a specific task that is assigned to them. They can be called within the main function or by other functions depending on how they are to be accessed.
A function definition or the way a function is designed contains few elements:

a. A return type: This defines the type of result the function would provide after the function is executed.
b. A function name: The function name is used to identify a function and as per standard coding guidelines the name of a function must reflect what the function does for better understanding.
c. A parameter list: These are values that are passed to the function and can be possibly empty.
d. A function body: The statements of a function that are enclosed within the curly braces.