CHAPTER 0: INTRODUCTION

C is a general-purpose programming language. It has been closely associated with the UNIX system, since it was developed on that system, and since UNIX and its software are written in C. The language, however, is not tied to any one operating system or machine; and although it has been called a "system programming language" because it is useful for writing operating systems, it has been used equally well to write major numerical, text-processing, and data-base programs.

C is a relatively "low level" language. This characterization is not pejorative; it simply means that C deals with the same sort of objects that most computers do, namely characters, numbers, and addresses. These may be combined and moved about with the usual arithmetic and logical operators implemented by actual machines.

C provides no operations to deal directly with composite objects such as character strings, sets, lists, or arrays considered as a whole. There is no analog, for example, of the PL/I operations which manipulate an entire array or string. The language does not define any storage allocation facility other than static definition and the stack discipline provided by the local variables of functions: there is no heap or garbage collection like that provided by Algol 68. Finally, C itself provides no input-output facilities: there are no READ or WRITE statements, and no wired-in file access methods. All of these higher-level mechanisms must be provided by explicitly-called functions.

The lack of a "heap" or "garbage collection" feature in C is both one of the great strengths of the language and at the same time is likely reason that the average programmer will never develop or maintain a major C application during their career.

C provides a simple feature using the `malloc()` and `free()` functions that allows a programmer to request a certain amount of memory be allocated dynamically, use the memory and then return the memory to the C runtime library for later reuse. For example to convert a JPG image to a PNG image, our application will read the JPG data into memory, then convert the image into a PNG image in memory and then write the PNG data out to a file. We don't know how large the images will be in advance, so we request whatever size we need from C and then give it back when we are done.

The term "heap" refers to memory that C manages on our behalf when we need to "borrow" a bit of memory and give it back later. There are a couple of issues with a simple heap implementation. First, if we "forget" to call `free()` when we are done with the memory, we have created a "memory leak" and our program eventually will run out of memory and abort. C places the onus of giving back any dynamically allocated memory on the programmer. Modern languages like Java, JavaScript, and Python keep track of when we stop using a segment of dynamic memory using a layer that can automatically reclaim the memory.