Safety and Strong Typing

Safety

One property that is sometimes looked for in a programming language is *safety*. There is some confusion as to what safety means, but here is a definition used in the year 2000 by an ad hoc committee that recommended that the Advanced Placement (AP) test in programming be based on the programming language Java:

> *Safety*: Any attempt to misinterpret data is caught at compile time or generates a well-specified error at runtime.

The programming languages C and C++ are not safe. An array index operation `b[k]` in these languages may be allowed even though `k` is outside the range of the array. There is also no built-in protection against using pointer manipulation to access almost any part of memory. "Buffer overflow" —akin to writing outside the boundary of an array— is exploited by many computer viruses. Of course, a programmer can explicitly check for overflow, but the language does not require it.

In the hands of a professional, the "unsafe" features of C and C++ can be used to advantage. But for most programmers, a language that provides safety can be of immense value. Especially for beginning programmers, safety helps.

Strong typing

Much (but not all) of safety is enforced by *strong typing* —this term, too, is not well defined, so let's explain what we mean by it here. First, the type of every variable and every expression is a syntactic property —known by looking at the program, without resorting to executing it. Second, a variable may be used only in ways that respect its type —for example, it is impossible to perform double operations on an int value (the int value must first be cast to double format), or to use a boolean value as an integer. Thus, syntactically, it is impossible to have an operation disregard the types of its operands, to interpret a value as something that it is not.

Runtime safety features

The strong typing of Java is backed up by strong guarantees on runtime behavior, the goal being to ensure that no value can be interpreted as something that it is not. With a `String` value, it is impossible to index a character of the `String` that does not exist. For example `s.charAt(i)`, will cause an exception if `i` is not in the range 0..`s.length`()-1. Similarly, array subscripts must be in the range of the array. Also, it is impossible to do "pointer arithmetic", as one can do in C and C++. In Java, the pointers are what we call the "names" of objects, and there are no operations on these names.

Benefits of safety and strong typing

Some errors arise from lack of understanding; others are logical errors caused by inadequate thinking and design; and some are simply typos. In all cases, finding errors early, at compile time, can save immense amounts of time. Safety and strong typing make possible the early detection of many errors.

The beginning programmer will benefit from safety and strong typing more than the professional programmer, simply because the beginner tends to make more of each type of error. But the professional programmer will also gain from a safe and strongly type language. When a program consists of hundreds of files and is being developed by a team of programmers, it is too easy to get confused and make mistakes, and the more checks we have on our programs, the better off we are.

Finally, with a language like Java, whose programs can be accessed on web pages (by writing *applets*), safety and strong typing aid the user as well, for these properties make it more difficult for malicious people to hack into systems through such applets.