Pointers

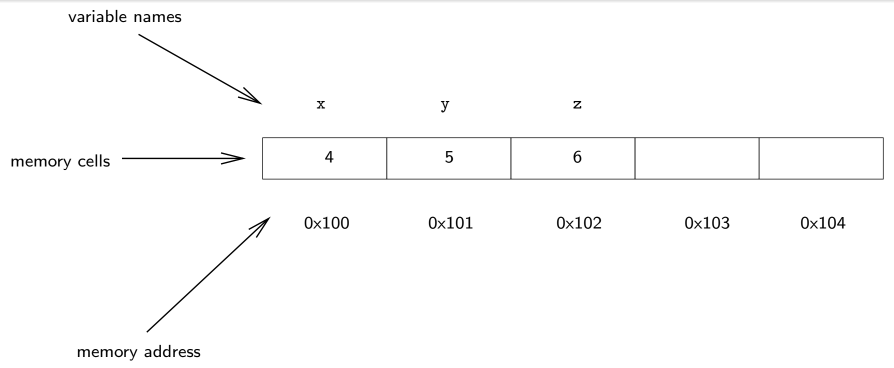
In this lecture we will focus on pointers and related aspects, such as:

- Strings
- Arrays
- Memory allocation

Memory

- Once we start looking at some code, keep in your mind that memory is nothing more than a long row of containers that can store bytes
  - We select a specific container by its address, which on a 32-bit CPU will look something like this `0xab23fe24`.
- In programming languages such as Java, and other similar languages, we get used to the idea that the compiler will manage the use of memory for us, allowing us to define and assign values to named variables, but we do not know exactly where in memory they are stored.
  - Indeed, we might mislead ourselves into believing that the computer is aware of our variable `date_of_birth` or `surname` — it's not, and it doesn't even know what a `string` is: all it knows are addresses and bytes.

Figure for Memory Use



Memory Allocation

- There are three ways of allocating memory within a process:
  - At compile time, hard-coded values (*e.g.* numbers, strings) will be stored somewhere in the executable file
  - The stack: local (*i.e.* temporary) variables of functions will be allocated on the stack when the function is called
  - The heap: the process may dynamically allocate (and later free) data on the heap, using functions such as `malloc` and `free`.

Study the Code

- A good understanding of pointers will be essential for the rest of the module, so:
  - Study the code in these examples
  - And try compiling and running the code, perhaps changing it to see what happens.