XB_40011 Style Guidelines

For the programs submitted to the assignments of this class, we mandate they follow the style guidelines as stated on this page. Doing so will be taken into account while grading.

We are fully aware that these guidelines are to some extent arbitrary and debatable. But any style guidelines would be. Not surprisingly(?), the zyBook does not follow *our* style guidelines. Later in your professional career, you will have to write code conforming to the style guidelines of the projects you are working with. Our guidelines are likely much simpler. Go ahead and get used to some guidance in the way you write code!

The following refers to concepts like functions and classes that you will learn about only later in the course. Please ignore them for now and come back to this page when you start using them.

Names

Meaningful names are the most important thing in programs. The names you give to the entities in your program (variables, objects, classes, functions,...) must clearly describe the purpose of these entities.

- For names that are combined from multiple words, we use **camelCase (https://en.wikipedia.org/wiki/Camel_case)** formatting.
- **Variables**: variables store "things". Their names should be nouns or noun phrases. They start with a lower case letter.
  - Examples: *elapsedTimeInDays*, *daysSinceModification*
- **Classes**: classes define "sorts of things". Their names should be nouns or noun phrases. They start with an upper case letter.
  - Examples: *Customer*, *Stack*, *Account*, *AddressParser*
- **Functions** and **methods**: functions and methods perform some "actions". Their names should have verb or verb phrase names. They start with a lower case letter.
  - Examples: *getName*, *isPosted*, *open*, *close*
- **Constants**: names of constant values (-> "things") are written completely in upper case letters. Instead of camelCase, we use underscores to separate nouns.
  - Examples: OVERTIME_RATE, WORKDAYS_PER_WEEK

Namespaces (e.g., std)

The use of *using namespace std* is not recommended but accepted. For example, we prefer using *std::cout* over just *cout*.

Indentation and Whitespace

We build programs by nesting small "boxes" into larger "boxes". Indentation represents exactly this nesting. Elements on the same nesting level must be equally indented.

We use whitespace to horizontally separate names from other characters like operators. We use empty lines to separate items (like functions) from each other.

Functions and Methods

Functions must not use any values other than their parameters. Read: *global variables are forbidden*. For methods, also the class members can be used.

Functions and methods should be as short and simple as possible. They should not mix concepts from multiple levels of abstraction. A function should do exactly one thing.

Const by default

Variables that are only initialised and not update later, should be declared as *const*. Rule of thumb: all variables should be declared as const, except those that, by design, are supposed to be changed at runtime. By analogy, the same holds for function parameters (when using references) and class methods.

Comments

Use comments only in exceptional cases where code needs explanation that can not be given by the code itself. Try to improve the names you use before writing a comment.

Consistency

These guidelines still leave a lot of freedom. One example is where to put opening and closing curly braces, { and }. When making choices, make sure that you are consistent with yourself. E.g., always position your { and } in the same places. This greatly simplifies the readability of your code. Keep in mind that readers always assume you will have a good reason if you write things in different ways.

Common sense

These style guidelines are to some extent idealized. You may occasionally run into a situation where deviating from these guidelines might be necessary. Feel free to do so. Such a case *might* require some comments in your code.