Preface

My interest in using LEGO® for teaching software internals goes back to Memory Dump Analysis Anthology¹, Volume 9b,² where I explained heap corruption, and illustrated linked lists, stacks, packed and unpacked structures. Later I applied the same technique for depicting trace and log analysis patterns³. My LEGO® modeling skills got a boost when I invented a baseplate representation of chemical structures⁴. Then I got an idea to represent machine learning topics using bricks⁵, and even abstract mathematics such as graphs and category theory⁶. After that, I recalled my usage of colored block diagrams to illustrate pointers⁷, Unified Modeling Language to represent Windows operating system internals such as drivers and their interaction⁸ and colored UML diagrams for debugging and diagnostic software construction patterns⁹. So all that fused into these series of short books (increments) you are reading now.

The first increment covers memory, memory addresses, pointers, program loading, kernel and user spaces, virtual process space, memory isolation, virtual and physical memory, memory paging, memory dump types.

² [https://www.dumpanalysis.org/Memory+Dump+Analysis+Anthology+Volume+9b](https://www.dumpanalysis.org/Memory+Dump+Analysis+Anthology+Volume+9b)
³ [https://www.dumpanalysis.org/lego-log-analysis](https://www.dumpanalysis.org/lego-log-analysis)
⁶ [https://www.dumpanalysis.org/visual-category-theory](https://www.dumpanalysis.org/visual-category-theory)
We represent computer memory as a linear sequence of memory cells.
Every cell contains some value. Empty cells contain 0 value. But in our representation, white cells may also contain unspecified values.
To identify every memory cell, we assign each cell a unique address.