About performance
Performance

• What is performance?
• How do we “add it to the program”?  
  • There is no silver bullet!
What is performance?
• Running “fast enough”
• Raw speed
• Responsiveness
  • Non-blocking
Improving performance

- Analyzing algorithms
- *Measuring execution time*
- Fixing algorithms
- Fine tuning the code
- Mastering memory manager
- *Writing parallel code*
- *Importing libraries*
Algorithm complexity
Algorithm complexity

- Tells us how algorithm slows down if data size is increased by a factor of n
- $O()$
  - $O(n)$, $O(n^2)$, $O(n \log n)$ ...
- Time **and** space complexity
• $O(1)$ accessing array elements
• $O(\log n)$ searching in ordered list
• $O(n)$ linear search
• $O(n \log n)$ quick sort (average)
• $O(n^2)$ quick sort (worst), naïve sort (bubblesort, insertion, selection)
• $O(cn)$ recursive Fibonacci, travelling salesman
## Comparing complexities

<table>
<thead>
<tr>
<th>Data size</th>
<th>$O(1)$</th>
<th>$O(\log n)$</th>
<th>$O(n)$</th>
<th>$O(n \log n)$</th>
<th>$O(n^2)$</th>
<th>$O(c^n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>4</td>
<td>10</td>
<td>43</td>
<td>100</td>
<td>512</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>8</td>
<td>100</td>
<td>764</td>
<td>10,000</td>
<td>$10^{29}$</td>
</tr>
<tr>
<td>300</td>
<td>1</td>
<td>9</td>
<td>300</td>
<td>2.769</td>
<td>90,000</td>
<td>$10^{90}$</td>
</tr>
</tbody>
</table>
Complexities in RTL

- Lists
  - access $O(1)$
  - search $O(n) / O(n \log n)$
  - sort $O(n \log n)$

- Dictionary
  - $*$ $O(1)$
  - search by value $O(n)$
  - Unordered!
    - Spring4D 1.2.1: CreateSortedDictionary

- Trees
  - $*$ $O(\log n)$
  - Spring4D 1.2.1: TRedBlackTree

http://bigocheatsheet.com/
Measuring performance
Measuring

- Manual
  - GetTickCount
  - QueryPerformanceCounter
  - TStopwatch

- Automated - Profilers
  - Sampling
  - Instrumenting
    - Binary
    - Source
Free profilers

- **AsmProfiler**
  - André Mussche
  - Instrumenting and sampling
  - 32-bit
  - [https://github.com/andremussche/asmprofiler](https://github.com/andremussche/asmprofiler)

- **Sampling Profiler**
  - Eric Grange
  - Sampling
  - 32-bit
  - [https://www.delphitools.info/samplingprofiler](https://www.delphitools.info/samplingprofiler)
Commercial profilers

- **AQTime**
  - Instrumenting and sampling
  - 32- and 64-bit
  - [https://smartbear.com/](https://smartbear.com/)

- **Nexus Quality Suite**
  - Instrumenting
  - 32- and 64-bit
  - [https://www.nexusdb.com](https://www.nexusdb.com)

- **ProDelphi**
  - Instrumenting (source)
  - 32- and 64-bit
  - [http://www.prodelphi.de/](http://www.prodelphi.de/)
A task for you!

If you choose to accept it ...
Task 1_1 Primoz\Task11

Move one line in function Test to a different place (in the same function) to make the code run faster.
Fixing the algorithm
Fixing the algorithm

• Find a better algorithm
• If a part of program is slow, don’t execute it so much
• If a part of program is slow, don’t execute it at all
“Don’t execute it so much”

- Don’t update UI thousands of times per second
- Call BeginUpdate/EndUpdate
- Don’t send around millions of messages per second
“Don’t execute it at all”

- UI virtualization
  - Virtual listbox
  - Virtual TreeView
- Memoization
  - Caching
  - Dynamic programming
  - TGpCache<K,V>
    - O(1) all operations
A task for you!

If you choose to accept it ...
Task 1_2 Primož\Task12

Make function Test run faster!
Fine tuning the code
## Compiler settings

![Compiler settings screenshot]

### Code generation
- **Code inlining control**: On
- **Code page**: 0
- **Emitter runtime type information**: False
- **Minimum enum size**: Byte
- **Optimization**: False
- **Pentium-safe FDIV**: False
- **Record field alignment**: Quad word
- **Stack frames**: True

### Debugging
- **Assertions**: True
- **Debug information**: Debug information
- **Local symbols**: True
- **Symbol reference info**: Reference info
- **Use debug dclcs**: True
- **Use imported data references**: True

### Other options
- **Runtime errors**: True
- **I/O checking**: True
- **Overflow checking**: False
- **Range checking**: False
- **Syntax options**:
Behind the scenes

- Strings
  - Reference counted, **Copy on write**

- Arrays
  - Static
  - Dynamic
    - Reference counted, Cloned

- Records
  - Initialized if managed

- Classes
  - Reference counted on ARC

- Interfaces
  - Reference counted
Calling methods

- Parameter passing
  - Dynamic arrays are strange

- Inlining
  - Single pass compiler!
A task for you!

If you choose to accept it ...
Task 1_3 Primoz\Task13

Without changing the program architecture, make it faster!
Memory management
Why memory manager?

- No fine-grained allocation on OS level
- Speed
- Additional functionality (debugging)
String and memory allocations

• NO†: string := string + ‘c’
• NO†: SetLength(array, Length(array) + 1)
• KIND OF OK‡: for ... do list.Add(something)

† KIND OF OK with a good memory manager
‡ May waste LOTS of memory [but improved in 10.3]
Memory management functions

- GetMem, AllocMem, ReallocMem, FreeMem
- GetMemory, ReallocMemory, FreeMemory
- New, Dispose
- Initialize, Finalize
Records vs. objects

• Objects
  • TObject.NewInstance
  • TObject.InstanceSize
  • GetMem
  • InitInstance
  • constructor (inherited ...)
  • AfterConstruction (inherited ...)

• Records†
  • GetMem
  • [_Initialize]

† Will change in 10.3
FastMM4: 58 memory managers in one!

Small block allocators
- 8-byte allocator
- 16-byte allocator
- Other allocators

Medium block allocator

Large block allocator

Image source: ‘Delphi High Performance’
© 2018 Packt Publishing
Optimizations

• Reallocation
  • Small blocks: New size = at least 2x old size
  • Medium blocks: New size = at least 1.25x old size

• Allocator locking
  • Small block only
  • Will try 2 ‘larger’ allocators
  • Problem: Freeing memory

allocIdx := find best allocator for the memory block
repeat
  if can lock allocIdx then
    break;
  Inc(allocIdx);
  if can lock allocIdx then
    break;
  Inc(allocIdx);
  if can lock allocIdx then
    break;
  Dec(allocIdx, 2)
until false
Optimizing parallel allocations

- FastMM4 from GitHub
  - https://github.com/pleriche/FastMM4
- DEFINE LogLockContention
  - or
- DEFINE UseReleaseStack
Alternatives

- ScaleMM
  - https://github.com/andremussche/scalemm

- TBBMalloc
  - https://www.threadingbuildingblocks.org
A task for you!

If you choose to accept it ...
Task 1_4 Primoz\Task14

Optimize memory allocations to make the program run faster!