**Eryk Ruciak K35 Projekt 3**

I. 4 metody sortowania tablicy dla liczb całkowitych (rzędu 50k  ‐  200k elementów) generowanych w 5 postaciach:

- postać losowa

|  |  |
| --- | --- |
| **Tablica losowa** | **t[s]** |
| **n** | **InsertionSort** | **SelectionSort** | **HeapSort** | **CocktailSort** |
| **50000** | **0,0001** | **1,5085** | **0,0028** | **0,0001** |
| **58000** | **0,0001** | **2,0069** | **0,0031** | **0,0001** |
| **66000** | **0,0001** | **2,6391** | **0,0036** | **0,0001** |
| **74000** | **0,0002** | **3,2588** | **0,0041** | **0,0001** |
| **82000** | **0,0002** | **3,8241** | **0,0047** | **0,0001** |
| **90000** | **0,0002** | **4,41241** | **0,005** | **0,0001** |
| **98000** | **0,0002** | **5,00072** | **0,0059** | **0,0001** |
| **106000** | **0,0009** | **5,58903** | **0,0058** | **0,0001** |
| **114000** | **0,0002** | **6,17734** | **0,0063** | **0,0001** |
| **122000** | **0,0003** | **6,76565** | **0,0068** | **0,0001** |
| **130000** | **0,0003** | **7,35396** | **0,0078** | **0,0002** |
| **138000** | **0,0003** | **7,94227** | **0,0116** | **0,0002** |
| **146000** | **0,0003** | **8,53058** | **0,011** | **0,0002** |
| **154000** | **0,0004** | **9,11889** | **0,0091** | **0,0003** |
| **162000** | **0,0003** | **9,7072** | **0,0095** | **0,0003** |
| **170000** | **0,0003** | **10,29551** | **0,0098** | **0,0004** |
| **178000** | **0,0006** | **10,88382** | **0,0105** | **0,0004** |
| **186000** | **0,0004** | **11,47213** | **0,0104** | **0,0004** |
| **194000** | **0,0004** | **12,06044** | **0,0115** | **0,0005** |

Wykresy:

- postać rosnąca

|  |  |
| --- | --- |
| **Tablica rosnąca** | **t[s]** |
| **n** | **InsertionSort** | **SelectionSort** | **HeapSort** | **CocktailSort** |
| **50000** | **0,0001** | **1,6164** | **0,0028** | **0,0001** |
| **58000** | **0,0001** | **2,1557** | **0,003** | **0,0001** |
| **66000** | **0,0001** | **2,8055** | **0,0036** | **0,0001** |
| **74000** | **0,0001** | **3,5067** | **0,004** | **0,0001** |
| **82000** | **0,0002** | **4,3219** | **0,0047** | **0,0001** |
| **90000** | **0,0002** | **4,90984** | **0,005** | **0,0002** |
| **98000** | **0,0002** | **5,58604** | **0,0055** | **0,0001** |
| **106000** | **0,0002** | **6,26224** | **0,0062** | **0,0001** |
| **114000** | **0,0003** | **6,93844** | **0,0068** | **0,0001** |
| **122000** | **0,0002** | **7,61464** | **0,0071** | **0,0001** |
| **130000** | **0,0003** | **8,29084** | **0,0076** | **0,0002** |
| **138000** | **0,0004** | **8,96704** | **0,0085** | **0,0001** |
| **146000** | **0,0003** | **9,64324** | **0,0084** | **0,0001** |
| **154000** | **0,0003** | **10,31944** | **0,0152** | **0,0001** |
| **162000** | **0,0004** | **10,99564** | **0,0095** | **0,0002** |
| **170000** | **0,0004** | **11,67184** | **0,0102** | **0,0002** |
| **178000** | **0,0004** | **12,34804** | **0,0105** | **0,0002** |
| **186000** | **0,0004** | **13,02424** | **0,0108** | **0,0002** |
| **194000** | **0,0004** | **13,70044** | **0,0116** | **0,0002** |

Wykresy:

- postać malejąca

|  |  |
| --- | --- |
| **Tablica malejąca** | **t[s]** |
| **n** | **InsertionSort** | **SelectionSort** | **HeapSort** | **CocktailSort** |
| **50000** | **0,0001** | **1,5159** | **0,0029** | **0,0001** |
| **58000** | **0,0001** | **2,0441** | **0,0034** | **0,0001** |
| **66000** | **0,0001** | **2,6639** | **0,0036** | **0,0001** |
| **74000** | **0,0001** | **3,341** | **0,0041** | **0,0001** |
| **82000** | **0,0002** | **3,915** | **0,0046** | **0,0001** |
| **90000** | **0,0002** | **4,52451** | **0,0052** | **0,0001** |
| **98000** | **0,0002** | **5,13402** | **0,0076** | **0,0001** |
| **106000** | **0,0002** | **5,74353** | **0,0093** | **0,0001** |
| **114000** | **0,0002** | **6,35304** | **0,0066** | **0,0001** |
| **122000** | **0,0003** | **6,96255** | **0,0071** | **0,0001** |
| **130000** | **0,0003** | **7,57206** | **0,0073** | **0,0001** |
| **138000** | **0,0003** | **8,18157** | **0,0078** | **0,0001** |
| **146000** | **0,0003** | **8,79108** | **0,0085** | **0,0003** |
| **154000** | **0,0003** | **9,40059** | **0,0091** | **0,0002** |
| **162000** | **0,0003** | **10,0101** | **0,0096** | **0,0002** |
| **170000** | **0,0004** | **10,61961** | **0,0104** | **0,0002** |
| **178000** | **0,0004** | **11,22912** | **0,0104** | **0,0002** |
| **186000** | **0,0004** | **11,83863** | **0,018** | **0,0003** |
| **194000** | **0,0004** | **12,44814** | **0,0112** | **0,0003** |

- postać stała

|  |  |
| --- | --- |
| **Stała** | **t[s]** |
| **n** | **InsertionSort** | **SelectionSort** | **HeapSort** | **CocktailSort** |
| **50000** | **0,0001** | **1,527** | **0,0003** | **0,0001** |
| **58000** | **0,0001** | **2,0228** | **0,0003** | **0,0001** |
| **66000** | **0,0001** | **2,5921** | **0,0004** | **0,0001** |
| **74000** | **0,0002** | **3,2571** | **0,0005** | **0,0001** |
| **82000** | **0,0002** | **3,78965** | **0,0005** | **0,0001** |
| **90000** | **0,0002** | **4,36561** | **0,0006** | **0,0001** |
| **98000** | **0,0002** | **4,94157** | **0,0007** | **0,0001** |
| **106000** | **0,0002** | **5,51753** | **0,0007** | **0,0001** |
| **114000** | **0,0002** | **6,09349** | **0,0007** | **0,0001** |
| **122000** | **0,0003** | **6,66945** | **0,0008** | **0,0001** |
| **130000** | **0,0003** | **7,24541** | **0,0009** | **0,0001** |
| **138000** | **0,0003** | **7,82137** | **0,0009** | **0,0001** |
| **146000** | **0,0003** | **8,39733** | **0,0009** | **0,0002** |
| **154000** | **0,0003** | **8,97329** | **0,001** | **0,0002** |
| **162000** | **0,0003** | **9,54925** | **0,001** | **0,0002** |
| **170000** | **0,0004** | **10,12521** | **0,001** | **0,0002** |
| **178000** | **0,0004** | **10,70117** | **0,0011** | **0,0002** |
| **186000** | **0,0004** | **11,27713** | **0,003** | **0,0002** |
| **194000** | **0,0004** | **11,85309** | **0,0023** | **0,0002** |

- postać v-kształtna:

|  |  |
| --- | --- |
| **V-ksztaltna** | **t[s]** |
| **n** | **InsertionSort** | **SelectionSort** | **HeapSort** | **CocktileSort** |
| **50000** | **0,0001** | **1,298** | **0,0061** | **0,0001** |
| **58000** | **0,0001** | **1,734** | **0,005** | **0,0001** |
| **66000** | **0,0001** | **2,2655** | **0,0037** | **0,0001** |
| **74000** | **0,0002** | **2,842** | **0,0042** | **0,0001** |
| **82000** | **0,0002** | **3,5379** | **0,0047** | **0,0008** |
| **90000** | **0,0002** | **4,2235** | **0,0051** | **0,0001** |
| **98000** | **0,0002** | **5,0061** | **0,0057** | **0,0001** |
| **106000** | **0,0002** | **5,81** | **0,0061** | **0,0001** |
| **114000** | **0,0003** | **6,8423** | **0,0065** | **0,0001** |
| **122000** | **0,0003** | **7,8513** | **0,0068** | **0,0001** |
| **130000** | **0,0003** | **8,129** | **0,0076** | **0,0001** |
| **138000** | **0,0003** | **8,85408** | **0,0079** | **0,0002** |
| **146000** | **0,0003** | **9,57916** | **0,0087** | **0,0002** |
| **154000** | **0,0003** | **10,30424** | **0,0091** | **0,0002** |
| **162000** | **0,0003** | **11,02932** | **0,0119** | **0,0002** |
| **170000** | **0,0003** | **11,7544** | **0,0108** | **0,0004** |
| **178000** | **0,0004** | **12,47948** | **0,0109** | **0,0002** |
| **186000** | **0,0004** | **13,20456** | **0,0116** | **0,0003** |
| **194000** | **0,0004** | **13,92964** | **0,0129** | **0,0002** |

II. Efektywność działania algorytmów

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **InsertionSort** | |  |  |  |  |
| **n** | **Malejąca** | **Losowa** | **Rosnąca** | **stała** | **v-ksztaltna** |
| **50000** | **0,0001** | **0,0001** | **0,0001** | **0,0001** | **0,0001** |
| **58000** | **0,0001** | **0,0001** | **0,0001** | **0,0001** | **0,0001** |
| **66000** | **0,0001** | **0,0001** | **0,0001** | **0,0001** | **0,0001** |
| **74000** | **0,0001** | **0,0002** | **0,0001** | **0,0002** | **0,0002** |
| **82000** | **0,0002** | **0,0002** | **0,0002** | **0,0002** | **0,0002** |
| **90000** | **0,0002** | **0,0002** | **0,0002** | **0,0002** | **0,0002** |
| **98000** | **0,0002** | **0,0002** | **0,0002** | **0,0002** | **0,0002** |
| **106000** | **0,0002** | **0,0009** | **0,0002** | **0,0002** | **0,0002** |
| **114000** | **0,0002** | **0,0002** | **0,0003** | **0,0002** | **0,0003** |
| **122000** | **0,0003** | **0,0003** | **0,0002** | **0,0003** | **0,0003** |
| **130000** | **0,0003** | **0,0003** | **0,0003** | **0,0003** | **0,0003** |
| **138000** | **0,0003** | **0,0003** | **0,0004** | **0,0003** | **0,0003** |
| **146000** | **0,0003** | **0,0003** | **0,0003** | **0,0003** | **0,0003** |
| **154000** | **0,0003** | **0,0004** | **0,0003** | **0,0003** | **0,0003** |
| **162000** | **0,0003** | **0,0003** | **0,0004** | **0,0003** | **0,0003** |
| **170000** | **0,0004** | **0,0003** | **0,0004** | **0,0004** | **0,0003** |
| **178000** | **0,0004** | **0,0006** | **0,0004** | **0,0004** | **0,0004** |
| **186000** | **0,0004** | **0,0004** | **0,0004** | **0,0004** | **0,0004** |
| **194000** | **0,0004** | **0,0004** | **0,0004** | **0,0004** | **0,0004** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SelectionsSort** | | | | | |
| **n** | **Malejąca** | **Losowa** | **Rosnąca** | **stała** | **v-ksztaltna** |
| **50000** | **1,5159** | **1,5085** | **1,6164** | **1,527** | **1,298** |
| **58000** | **2,0441** | **2,0069** | **2,1557** | **2,0228** | **1,734** |
| **66000** | **2,6639** | **2,6391** | **2,8055** | **2,5921** | **2,2655** |
| **74000** | **3,341** | **3,2588** | **3,5067** | **3,2571** | **2,842** |
| **82000** | **3,915** | **3,8241** | **4,3219** | **3,78965** | **3,5379** |
| **90000** | **4,52451** | **4,41241** | **4,90984** | **4,36561** | **4,2235** |
| **98000** | **5,13402** | **5,00072** | **5,58604** | **4,94157** | **5,0061** |
| **106000** | **5,74353** | **5,58903** | **6,26224** | **5,51753** | **5,81** |
| **114000** | **6,35304** | **6,17734** | **6,93844** | **6,09349** | **6,8423** |
| **122000** | **6,96255** | **6,76565** | **7,61464** | **6,66945** | **7,8513** |
| **130000** | **7,57206** | **7,35396** | **8,29084** | **7,24541** | **8,129** |
| **138000** | **8,18157** | **7,94227** | **8,96704** | **7,82137** | **8,85408** |
| **146000** | **8,79108** | **8,53058** | **9,64324** | **8,39733** | **9,57916** |
| **154000** | **9,40059** | **9,11889** | **10,31944** | **8,97329** | **10,30424** |
| **162000** | **10,0101** | **9,7072** | **10,99564** | **9,54925** | **11,02932** |
| **170000** | **10,61961** | **10,29551** | **11,67184** | **10,12521** | **11,7544** |
| **178000** | **11,22912** | **10,88382** | **12,34804** | **10,70117** | **12,47948** |
| **186000** | **11,83863** | **11,47213** | **13,02424** | **11,27713** | **13,20456** |
| **194000** | **12,44814** | **12,06044** | **13,70044** | **11,85309** | **13,92964** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **HeapSort** | | | | | |
| **n** | **Malejąca** | **Losowa** | **Rosnąca** | **stała** | **v-ksztaltna** |
| **50000** | **0,0029** | **0,0028** | **0,0028** | **0,0003** | **0,0061** |
| **58000** | **0,0034** | **0,0031** | **0,003** | **0,0003** | **0,005** |
| **66000** | **0,0036** | **0,0036** | **0,0036** | **0,0004** | **0,0037** |
| **74000** | **0,0041** | **0,0041** | **0,004** | **0,0005** | **0,0042** |
| **82000** | **0,0046** | **0,0047** | **0,0047** | **0,0005** | **0,0047** |
| **90000** | **0,0052** | **0,005** | **0,005** | **0,0006** | **0,0051** |
| **98000** | **0,0076** | **0,0059** | **0,0055** | **0,0007** | **0,0057** |
| **106000** | **0,0093** | **0,0058** | **0,0062** | **0,0007** | **0,0061** |
| **114000** | **0,0066** | **0,0063** | **0,0068** | **0,0007** | **0,0065** |
| **122000** | **0,0071** | **0,0068** | **0,0071** | **0,0008** | **0,0068** |
| **130000** | **0,0073** | **0,0078** | **0,0076** | **0,0009** | **0,0076** |
| **138000** | **0,0078** | **0,0116** | **0,0085** | **0,0009** | **0,0079** |
| **146000** | **0,0085** | **0,011** | **0,0084** | **0,0009** | **0,0087** |
| **154000** | **0,0091** | **0,0091** | **0,0152** | **0,001** | **0,0091** |
| **162000** | **0,0096** | **0,0095** | **0,0095** | **0,001** | **0,0119** |
| **170000** | **0,0104** | **0,0098** | **0,0102** | **0,001** | **0,0108** |
| **178000** | **0,0104** | **0,0105** | **0,0105** | **0,0011** | **0,0109** |
| **186000** | **0,018** | **0,0104** | **0,0108** | **0,003** | **0,0116** |
| **194000** | **0,0112** | **0,0115** | **0,0116** | **0,0023** | **0,0129** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CocktailSort** | | | | | |
| **n** | **Malejąca** | **Losowa** | **Rosnąca** | **stała** | **v-ksztaltna** |
| **50000** | **0,0001** | **0,0001** | **0,0001** | **0,0001** | **0,0001** |
| **58000** | **0,0001** | **0,0001** | **0,0001** | **0,0001** | **0,0001** |
| **66000** | **0,0001** | **0,0001** | **0,0001** | **0,0001** | **0,0001** |
| **74000** | **0,0001** | **0,0001** | **0,0001** | **0,0001** | **0,0001** |
| **82000** | **0,0001** | **0,0001** | **0,0001** | **0,0001** | **0,0008** |
| **90000** | **0,0001** | **0,0001** | **0,0002** | **0,0001** | **0,0001** |
| **98000** | **0,0001** | **0,0001** | **0,0001** | **0,0001** | **0,0001** |
| **106000** | **0,0001** | **0,0001** | **0,0001** | **0,0001** | **0,0001** |
| **114000** | **0,0001** | **0,0001** | **0,0001** | **0,0001** | **0,0001** |
| **122000** | **0,0001** | **0,0001** | **0,0001** | **0,0001** | **0,0001** |
| **130000** | **0,0001** | **0,0002** | **0,0002** | **0,0001** | **0,0001** |
| **138000** | **0,0001** | **0,0002** | **0,0001** | **0,0001** | **0,0002** |
| **146000** | **0,0003** | **0,0002** | **0,0001** | **0,0002** | **0,0002** |
| **154000** | **0,0002** | **0,0003** | **0,0001** | **0,0002** | **0,0002** |
| **162000** | **0,0002** | **0,0003** | **0,0002** | **0,0002** | **0,0002** |
| **170000** | **0,0002** | **0,0004** | **0,0002** | **0,0002** | **0,0004** |
| **178000** | **0,0002** | **0,0004** | **0,0002** | **0,0002** | **0,0002** |
| **186000** | **0,0003** | **0,0004** | **0,0002** | **0,0002** | **0,0003** |
| **194000** | **0,0003** | **0,0005** | **0,0002** | **0,0002** | **0,0002** |

III. Implementacja algorytmu Quicksort w dwóch wersjach: rekurencyjnie i iteracyjnie oraz porównywanie ich:

-QuickSort Iteracyjnie

Wyniki:

|  |  |
| --- | --- |
| **n** | **t[s]** |
| **50000** | 0,0031 |
| **54000** | 0,003 |
| **58000** | 0,0032 |
| **62000** | 0,0034 |
| **66000** | 0,0047 |
| **70000** | 0,0046 |
| **74000** | 0,0046 |
| **78000** | 0,0052 |
| **82000** | 0,0052 |
| **86000** | 0,0054 |
| **90000** | 0,0057 |
| **94000** | 0,0057 |
| **98000** | 0,0059 |
| **102000** | 0,0064 |
| **106000** | 0,0065 |
| **110000** | 0,0067 |
| **114000** | 0,007 |
| **118000** | 0,007 |
| **122000** | 0,0074 |
| **126000** | 0,0077 |
| **130000** | 0,0077 |
| **134000** | 0,0103 |
| **138000** | 0,0107 |
| **142000** | 0,011 |
| **146000** | 0,0108 |
| **150000** | 0,0113 |
| **154000** | 0,0125 |
| **158000** | 0,0114 |
| **162000** | 0,0116 |
| **166000** | 0,0118 |
| **170000** | 0,0116 |
| **174000** | 0,0121 |
| **178000** | 0,012 |
| **182000** | 0,0125 |
| **186000** | 0,0122 |
| **190000** | 0,0125 |
| **194000** | 0,0126 |
| **198000** | 0,0131 |

- QuickSort Rekurencyjnie

Wyniki:

|  |  |
| --- | --- |
| **n** | **t[s]** |
| **50000** | 0,0372 |
| **54000** | 0,0377 |
| **58000** | 0,0372 |
| **62000** | 0,038 |
| **66000** | 0,0746 |
| **70000** | 0,0735 |
| **74000** | 0,0735 |
| **78000** | 0,074 |
| **82000** | 0,0724 |
| **86000** | 0,0729 |
| **90000** | 0,0728 |
| **94000** | 0,0727 |
| **98000** | 0,072 |
| **102000** | 0,0742 |
| **106000** | 0,0752 |
| **110000** | 0,0734 |
| **114000** | 0,0738 |
| **118000** | 0,0751 |
| **122000** | 0,0753 |
| **126000** | 0,074 |
| **130000** | 0,0746 |
| **134000** | 0,1447 |
| **138000** | 0,1457 |
| **142000** | 0,1462 |
| **146000** | 0,1453 |
| **150000** | 0,1451 |
| **154000** | 0,145 |
| **158000** | 0,1456 |
| **162000** | 0,1462 |
| **166000** | 0,1486 |
| **170000** | 0,1464 |
| **174000** | 0,1472 |
| **178000** | 0,1474 |
| **182000** | 0,1472 |
| **186000** | 0,1475 |
| **190000** | 0,149 |
| **194000** | 0,1481 |
| **198000** | 0,1471 |

**Wnioski:**

1. Sortowanie przez wybieranie jest jedną z prostszych, dlatego również jedną z częstych metod sortowań. Wyszukuje ona element, który ma znajdować się na danej pozycji a następnie zamienia go z tam obecnym. Operacja ta wykonywana jest dla każdego indeksu dlatego sortowanie te trwa tak długo. W związku z tym aby przyśpieszyć i pobrać mniej pamięci komputera do trudniejszych sortowań stosuje się inne metody.

2. Sortowanie przez wstawianie, również ma prostą zasadę działania i nadaję się najlepiej do posortowania tablicy o niewielkiej liczbie elementów.

3. Sortowanie stogowe działa szybciej niż dwa poprzednie oraz zużywa mało pamięci, niestety jednak jest niestabilne. Sortowanie te jest nieco wolniejsze od sortowania szybkiego, lecz ma lepszą pesymistyczną złożoność czasową.

5. Sortowanie koktajlowe jest najstabilniejszym sposobem sortowania, czas wykonywania również nie jest długi, polega na sortowaniu za pomocą porównań.

6. Sortowanie szybkie przez iterację odbywa się szybciej od rekurencyjnego sortowania, ponieważ nie wywołuje ono każdego elementu z osobna.

Kod źródłowy:

const int NIter = 10;

Console.WriteLine("n\tt[s]");

for (int i = 50000; i <= 200000; i += 8000)

{

int[] tablica = new int[i];

Random(tablica);

double ElapsedSeconds;

long ElapsedTime = 0, MinTime = long.MaxValue, MaxTime = long.MinValue, IterationElapsedTime;

for (int n = 0; n < (NIter + 1 + 1); ++n)

{

long StartingTime = Stopwatch.GetTimestamp();

CocktailSort(tablica);

long EndingTime = Stopwatch.GetTimestamp();

IterationElapsedTime = EndingTime - StartingTime;

ElapsedTime += IterationElapsedTime;

if (IterationElapsedTime < MinTime) MinTime = IterationElapsedTime;

if (IterationElapsedTime > MaxTime) MaxTime = IterationElapsedTime;

}

ElapsedTime -= (MinTime + MaxTime);

ElapsedSeconds = ElapsedTime \* (1.0 / (NIter \* Stopwatch.Frequency));

Console.WriteLine("{0}\t{1}", i, ElapsedSeconds.ToString("F4"));

private static void Random(int[] tablica)

{

Random rnd = new Random(Guid.NewGuid().GetHashCode());

for (int i = 0; i < tablica.Length; i++)

{

tablica[i] = rnd.Next(int.MaxValue);

}

private static void CocktailSort(int[] t)

{

int Left = 1, Right = t.Length - 1, k = t.Length - 1;

do

{

for (int j = Right; j >= Left; j--)

if (t[j - 1] > t[j])

{

int Buf = t[j - 1]; t[j - 1] = t[j]; t[j] = Buf;

k = j;

}

Left = k + 1;

for (int j = Left; j <= Right; j++)

if (t[j - 1] > t[j])

{

int Buf = t[j - 1]; t[j - 1] = t[j]; t[j] = Buf;

k = j;

}

Right = k - 1;

}

while (Left <= Right);

private static void Heapify(int[] t, uint left, uint right)

{

uint i = left,

j = 2 \* i + 1;

int buf = t[i];

while (j <= right)

{

if (j < right)

if (t[j] < t[j + 1]) j++;

if (buf >= t[j]) break;

t[i] = t[j];

i = j;

j = 2 \* i + 1;

}

t[i] = buf;

}

private static void HeapSort(int[] t)

{

uint left = ((uint)t.Length / 2),

right = (uint)t.Length - 1;

while (left > 0)

{

left--;

Heapify(t, left, right);

}

while (right > 0)

{

int buf = t[left];

t[left] = t[right];

t[right] = buf;

right--;

Heapify(t, left, right);

}

}

}

}

private static void InsertionSort(int[] t)

{

for (uint i = 1; i < t.Length; i++)

{

uint j = i;

int Buf = t[j];

while ((j > 0) && (t[j - 1] > Buf))

{

t[j] = t[j - 1];

j--;

}

t[j] = Buf;

private static void SelectionSort(int[] t)

{

uint k;

for (uint i = 0; i < (t.Length - 1); i++)

{

int Buf = t[i];

k = i;

for (uint j = i + 1; j < t.Length; j++)

if (t[j] < Buf)

{

k = j;

Buf = t[j];

}

t[k] = t[i];

t[i] = Buf;

}

private static void Growing(int[] tablica)

{

for (int i = 0; i < tablica.Length; i++)

{ tablica[i] = i; }

}

private static void Descending(int[] tablica)

{

int x = tablica.Length;

for (int i = 0; i < tablica.Length; i++)

{

tablica[i] = x;

x--;

private static void Constantly(int[] tablica)

{

for (int i = 0; i < tablica.Length; i++)

{ tablica[i] = 10; }

}

private static void VShaped(int[] tablica)

{

int y = tablica.Length;

int z = 1;

for (int i = 0; i <= (tablica.Length / 2) - 1; i++)

{

tablica[i] = y;

y -= 2;

}

for (int i = tablica.Length / 2; i < tablica.Length; i++)

{

tablica[i] = z;

z += 2;

}

Eryk Ruciak