Prof. Dr. A.K. Hartmann
WiSe 2023/24, 5.04.4521
Blatt 8, 27.02.2024

Exercises Computational Physics

8 Event-driven simulations

1. Download the program `chain_heap_fragment.c` from `StudIP`.
2. Investigate the so-far program thoroughly. In particular have a look at
   - The data structures `particle_t`, `event_t`, `heap_elem_t` and `global_t`
   - The implementation of the `main()` function
   - The prototypes (parameters) of all functions (read the comments above the function)
   - The implementation of the function `treat_event()`
   - The implementation of the function `heap_remove()`, which removes one element from the heap. Keep in mind that whenever an event element changes its position in the heap, the reference `event[...].pos` of the element in the 'event' array has to be updated.

   Hint: the programm can be compiled without other source files (only flag `-lm` strictly necessary) and is called by

   ```
   chain_heap_fragment <#particles> <t_max>
   ```
3. Implement the function

   ```
   /******************** heap_insert() *****************/
   /** Insert event 'event[ev]' into heap 'glob->heap' **/
   /**  (and stores position in heap in event          **/
   /** PARAMETERS: (*)= return-paramter                **/
   /**       glob: global data                         **/
   /**      event: array of events                     **/
   /*          ev: id of event                         **/
   /** RETURNS:                                        **/
   /**    nothing                                      **/
   /*****************************************************/
   void heap_insert(global_t *glob, event_t *event, int ev)
   ```

   by compliting th so-far given function. Hints: in `heap_remove()` you can see how the (more complicated) removal of an element from the heap works and use it as blueprint. Keep in mind that the element `pos` is updated for *all* events which change heap position. (6 P)

4. Test the function by following the execution within the debugger `gdb` how an element is inserted.

   For this purpose, set a break point in the first line of the function, and then execute the function step by step once the breakpoint is reached. Print content of suitable variables, to check whether the execution is correct.

   (1 P)

5. Perform simulations and measure the densities for 1000 particles and final time $t = 10^5$ for

   a) particles with alternating masses 1 and 2.4 (like in lecture)

   b) particles with alternating masses 1 and 100

   c) particles with alternating masses 1 and 1.1

   d) particles with the same mass for all

   e) particles with random masses in $[1, 10]$

   (Which function you have to modify for b), c), d), e) m ?)

   and plot the resulting dsenity curves, e.g., with `gnuplot`. (3 P)