C++ course – Exercises Set 4

Wouter Verkerke (Jan 2023)

Exercise 4.1 – The Calorimeter example

The goal of this exercise is the implement the classes of the calorimeter example of lecture module 4, i.e. to implement the classes `Point`, `CaloCell`, `CaloGrid` and `Calorimeter`. *(These classes will be revisited later in the course to add further functionality)*

- The general approach for each of the 4 classes (`Point`, `CaloCell`, `CaloGrid` and `Calorimeter`), *is to first write the interface*, and only then the implementation.
    - Write a small main program that allows you to create an object of each new type you write to be able to test it
    - Write the classes in the order listed in this exercise

- **Class** `CaloCell`
    - Supply the data members `double energy` and `int ID`
    - The constructor should take as arguments the initial energy and the ID. *Do you need a copy constructor? Do you need a Destructor? If yes, implement them.*
    - Write accessors and modifiers for data members `energy`, `ID`
    - Think about which member functions should be `const` and make those functions `const`
    - Before proceeding to the next class, test your code by creating and using a `CaloCell` object in `main()`

- **Class** `Point`
    - Supply the data members `double x,y,z`
    - The constructor should take as arguments the initial values of `x,y,z` with default of (0,0,0).
      Do you need a copy constructor? Do you need a destructor? If yes, implement them.
    - Write accessors and modifiers for the individual data members `x,y,z`
    - Also add an additional modifier function that sets `x,y,z` in a single call.
    - Think about which member functions should be `const` and make those `const`
    - Before proceeding to the next class test your code by creating and using a `Point` object in `main()`

- **Class** `CaloGrid`
  - Supply the data members `int nx, ny` to hold the grid dimensions and a data member to hold a one-dimensional array of `CaloCell` elements sixed `nx*ny`. *Can you implement the array directly as data member, or does it need to be held in an external memory allocation with a data member that points to it?*
  - Write a constructor that takes the size in directions x and y as arguments. *Do you need a copy constructor? Do you need a Destructor? If yes, implement them.*
  - Test your code at this point. You may get compiler errors about the instantiation of an array of `CaloCells` objects. *What is the requirement on the constructor(s) of a class if you want to be able to allocate it as an array?* Modify the `CaloCell` constructor to make it work.
  - Add a member function of `CaloCell* cell(int x, int y)` that returns a cell for a given coordinate. *Check that the given x and y values are inside the range of your calorimeter grid.* If they are outside return a null pointer.
  - Add a function `const CaloCell* cell(int x, int y) const`. Can you implement this function by calling the non-constant version of this method?
  - Test your code again

- **Class** `Calorimeter`
  - Add two data members: a `CaloGrid` and a `Point`
  - Implement a constructor function that takes the dimensions of calorimeter and the initial space coordinates of its position as function arguments and pass those values to the constructor calls of the `CaloGrid` and `Point` data members.
  - *Do you need a copy constructor? Do you need a Destructor? If yes, implement them.*
  - Modify the constructor so that it provides a default value for the space coordinates.
  - Add the following accessors and modifiers:

```
CaloGrid& grid(),
const CaloGrid& grid() const,
Point& position() and
const Point& position() const
```

  - Do you need a copy constructor? Do you need a destructor?