B4 - Object-Oriented Programming

B-OOP-400

Return to C++

NanoTekSpice Bootstrap

2.0

{EPITECH.}

Electronic Simulator

The **NanoTekSpice** project being a digital electronic simulator based on a graph, the first thing you need to do is to discover what digital electronic is…

Here is an online digital electronic simulator. It lets you add components, link them together and run simulations. What you are trying to recreate is similar to this, with some additional real components.

There are three states in digital electronics: `true`, `false` and `undefined`. On **logic.ly**, `true` makes wires appear in blue, `undefined` makes them appear in grey and `false` in white.

`undefined` is the default state of every part of a circuit. Input states may make `undefined` states disappear if the logic if the circuit allows it.

In the end, this is all very similar to normal variables. Here are a few examples:

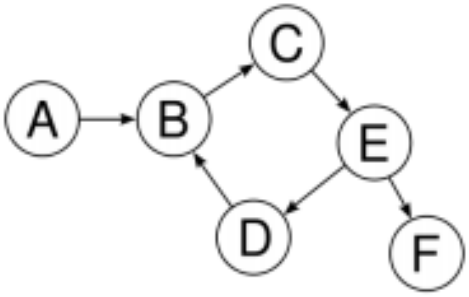
```
{
        bool a = true;
        bool b;
        bool c;

        c = a && b;
        // c is undefined because b is undefined and a is true, so the result of c
            depends only on b
}

{
        bool a = false;
        bool b;
        bool c;

        c = a && b;
        // c is false, because false AND anything is always false
}

{
        bool a = true;
        bool b;
        bool c;

        c = a || b;
        // c is true because true OR anything is always true
}

{
        bool a = false;
        bool b;
        bool c;

        c = a || b;
        // c is undefined because b is undefined and a is false, so the result of c
            depends only on b
}
```

+ CREATE A GRAPH

A graph is a data structure in which nodes are linked together, without any specific shape.

- A tree has a specific shape: there is always a root node with children, each of these children being the root of other children…
- A linked list has a specific shape: there is one node, linked to another, which may be linked again, and so on and so forth.

A graph's only requirement is that nodes are linked together. Therefore, trees and lists are two subsets of graphs, just like lists are a subset of trees.



Graphs can be oriented, meaning that links are unidirectional. In the example above, A is linked to B but B is not linked to A. Note that nodes can also be linked to themselves.

- Declare and implement the `Component` class which:
  - represents a component with inputs and outputs
  - may hold states
  - can be built with a value
  - can be executed and return a result
  - can be linked to other `Components`
- Declare and implement the `Circuit` class. It must manipulate `Components` and you must be able to:
  - add and remove components to the `Circuit`
  - run a simulation
  - use a `Circuit` as a `Component`
- Declare and implement the `Parser` class. It must manipulate a `Circuit` and:
  - take as parameters a file path and a `Circuit` to fill
  - throw exceptions of the according type when faced with an error

Although a graph is a container, we strongly recommend using standard containers to make implementing your graph easier.