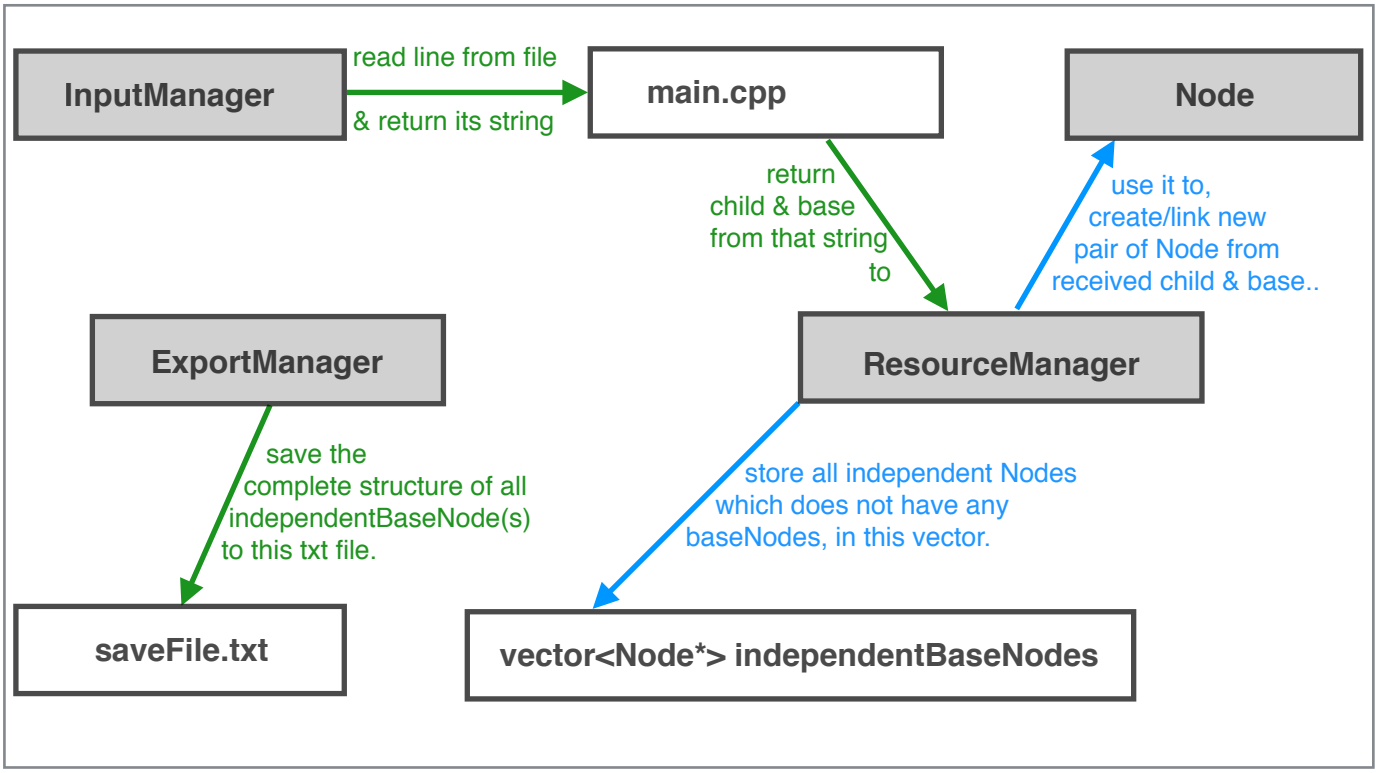
Assignment 1: Resource Manager for a Real Time Strategy (RTS) game (link)

Use make command to compile the code. This program is developed for terminal's g++ compiler.

Structure of Node:

Node	
string name	: stores name
vector<Node *> baseNode	: points to all baseNode(s)
vector<Node *> childNode	: points to all childNode(s)
vector<string> reliesOnNodes	: store name of all baseNodes on which it relies to be "usable"
bool usable	: TRUE, if all reliesOnNodes are there in baseNode in "usable" state

How the whole system is organised:



Syntax to add node/link:

- nameOfNode - add this node to independentBaseNodes vector.
- childNode baseNode - add link between these two nodes; if the node does not exist, it will create new node by the given name.

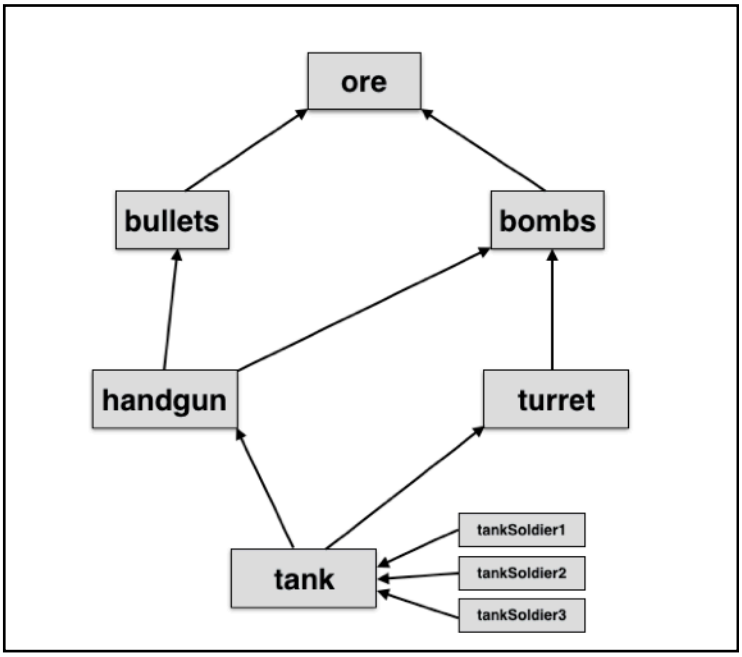
(Note: Memory management is also implemented when a delete action takes place..)

When aNode gets deleted:

- Reference of aNode will be removed from all of its immediate childNode(s).
- All childNode(s) of the deleted node will be recursively marked as “not-usable”.
- If a childNode does not have any other baseNode; that childNode will be added to independentBaseNodes vector.
- To make a childNode usable again, deleted node should be added and linked again as it was previously linked to the childNode.

How graph is plotted on the console:

- To create structure of dependencies like this one ->



- resource.txt can be like,

```
tankSoldier1 tank
tankSoldier2 tank
tankSoldier3 tank
turret
tank turret
tank handgun
turret bombs
handgun bullets
handgun bombs
bombs ore
bullets ore
```

- And output will be:

```
--------- Structure of "ore" ---------
||ore (usable):
     <-bombs (usable):
          <-turret (usable):
               <-tank (usable):
                    <-tankSoldier1 (usable)
                    <-tankSoldier2 (usable)
                    <-tankSoldier3 (usable)
          <-handgun (usable):
               <-tank (usable) ~link~
     <-bullets (usable):
          <-handgun (usable) ~link~

--------- ----------------- ---------
```

link to same node

~link~: denotes a link to the same Node which is already plotted once in the output of current structure.