

void isCircle()

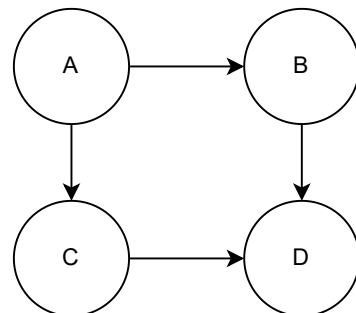
Создаем set с адресами Cell  
std::unordered\_set<Cell\*> counter для фиксации уже посещенных ячеек  
и std::unordered\_map<Cell\*, bool> reStack для фиксации пройденного пути  
графа

добавляем в set текущую ячейку  
counter.insert(Cell\*) и в reStack для  
текущей ячейки ставим true

Для каждой ссылки, на которую тек. ячейка ссылается рекурсивно вызываем  
помощника, передавая ему ссылку на сет счетчика и reStack

```
for(const Cell* cell : std::vector<Cell*> ref_cells){  
    if(!counter.contains(&(*cell))){  
        counter.insert(cell);  
        reStack[cell] = true;  
        for (const Cell* cell_d : cell.GetRefCells()){  
            if(!counter.contains(&(*cell_d)) && cell -> CircularDependency(&counter,  
                &reStack){  
  
                return true;  
            }else if (reStack[cell_d] == true){  
                throw CircleError ;  
            }  
        }  
    }  
}
```

```
reStack[cell_d] == false;  
return;
```



1. Создаем counter
2. В него добавляем ячейку A и в reStack ставим true
3. A содержит vector<Cell\*> {B,C}
4. Рекурсивно вызываем помощника у B
5. Проверяем, что его нет в counter и добавляем его и в reStack ставим true
6. У B vector<Cell\*> {D}
7. Вызываем рек-но помощника и D
8. Проверяем в counter, добавляем и ставим true.
9. Так как у D нет больше ссылок, то для него ставим false в reStack и возвращаемся из рекурсии
9. Рекурсивно вызываем помощника у C
10. Проверяем, добавляем и ставим true
11. У C vector<Cell\*> {D}
12. Запускаем у D
13. Он уже есть в counter, но не включен в путь. Оставляем ему false и возвращаемся