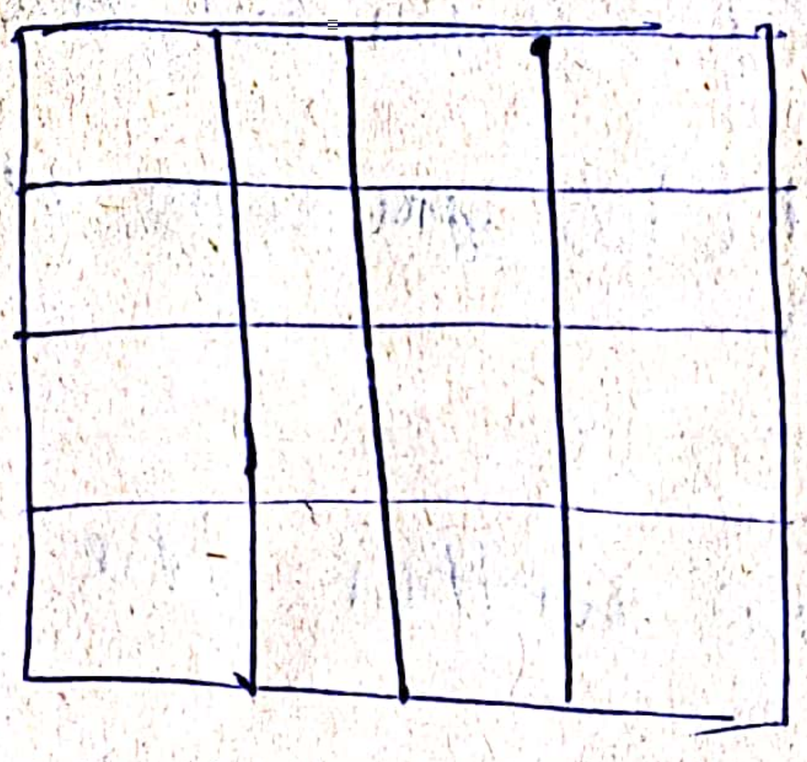
2048 Game:-



* we will be given a 4x4 grid which have some numbers.
* those numbers are always powers of 2
* we can perform the following operations.
  1) UP
  2) down
  3) left
  4) right
* you can play this game at the following website. "2048games.com". to understand the game.
* in this part we are going to code this game in C++.
* what are the function we need to make this game.
  1) UP() to move those elements upward
  2) down()

3) left()

4) Right()

5) is_game_over() // you lost the game

6) is_game_won() // you won the game

7) there can be more.

is_game_over() :-

* how do we know wether the game is over or not..
* in this state all the cells in the grid are filled ——→ ①
* you ~~shall~~ will not be able to perform any move (UP(), down(), left(), Right()). ——→ ②.
* so, we need to design the function in such a way thit it need to satisfy the both statement ① & ②.

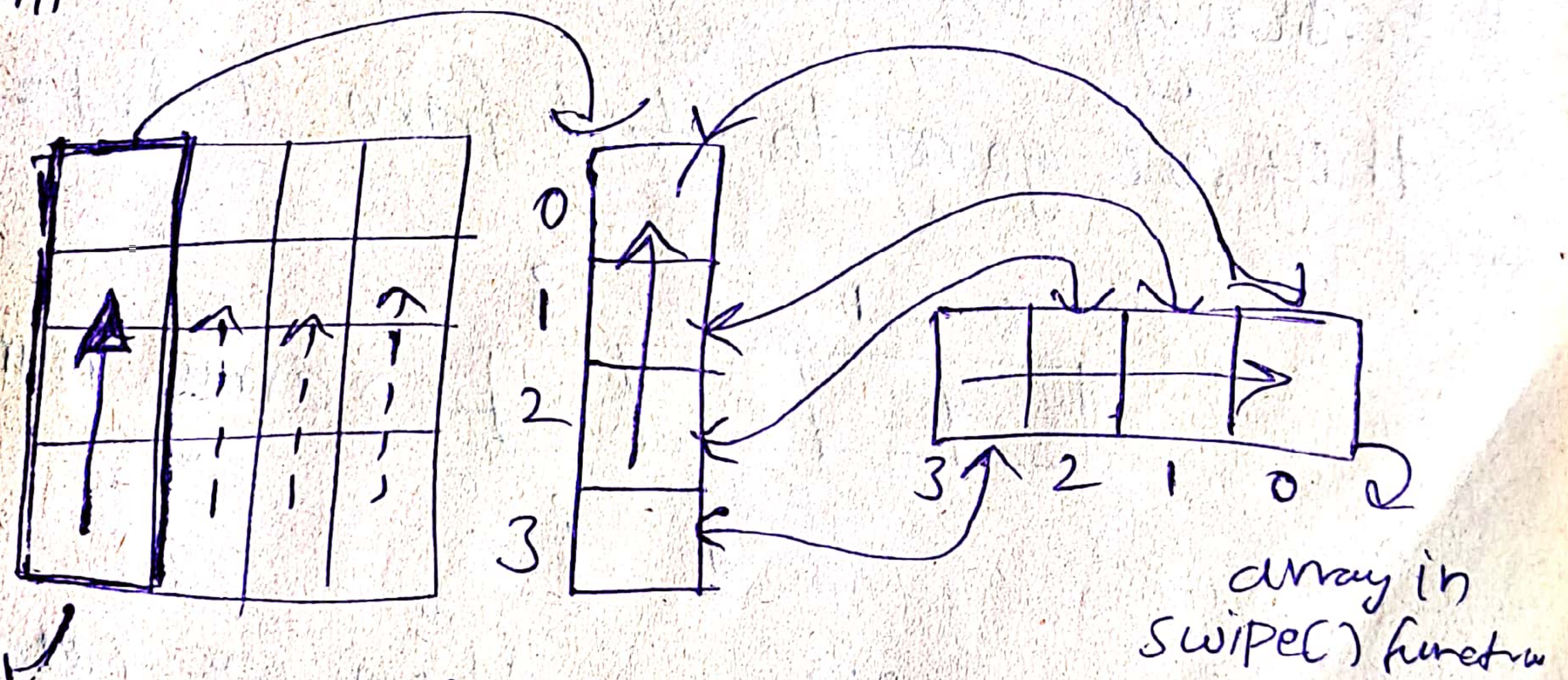
is_game_won() :-

* this is the most easiest function of this code, becuse you just need to cheek wether any cell contains 2048..

```
bool is_game_won(){
    for(int i=0;i<4;i++)
        {for(int j=0;j<4;j++){
            if(grid[i][j]==2048){
                return true;
            }
        }
    }
return false; }
```

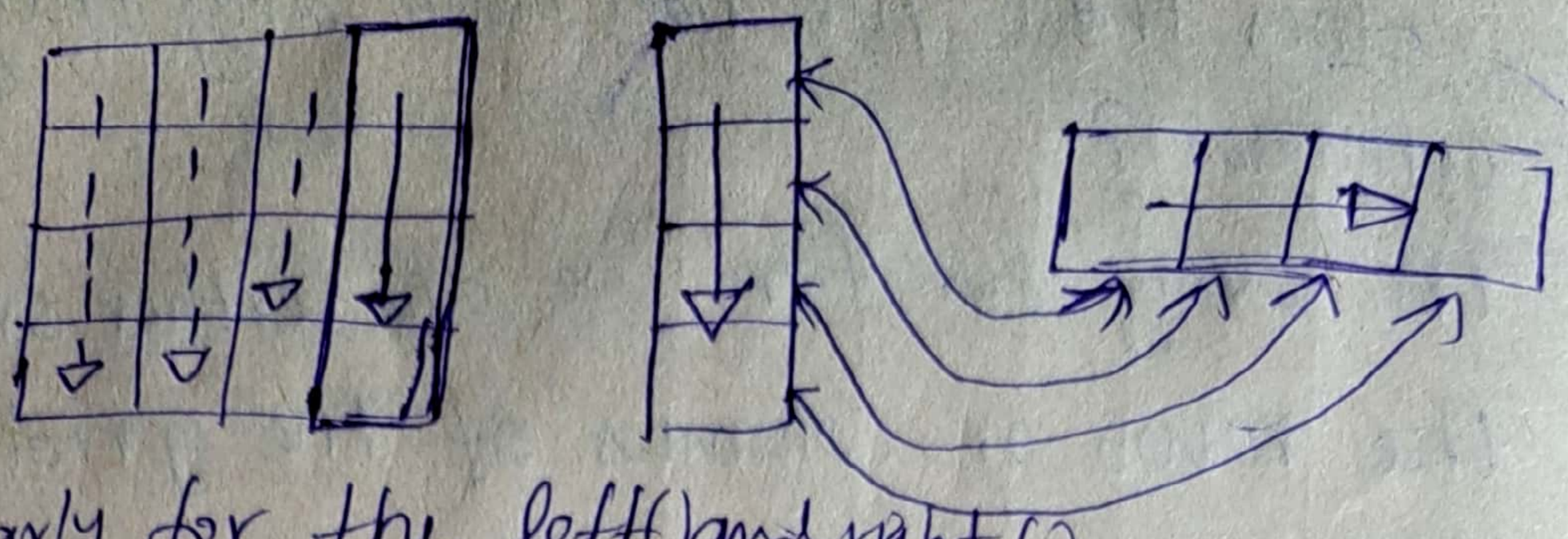
left(), Right(), UP(), down():-

* does these function have any thing in common?
* in every function we are either swiping the rows or columns of the grid in a direction..
* it will be easy for us to make a function that takes a specific row/col and moves/swipes them to the right.
* all we need to do is how to pass a row/column and take that back from the swipe function
* we create the swipe function in such a way that it always swipes the to the right
*



here we are performing an UP() operation on the grid, so, we pass each column to the swipe() using for loop.

Similarly down()



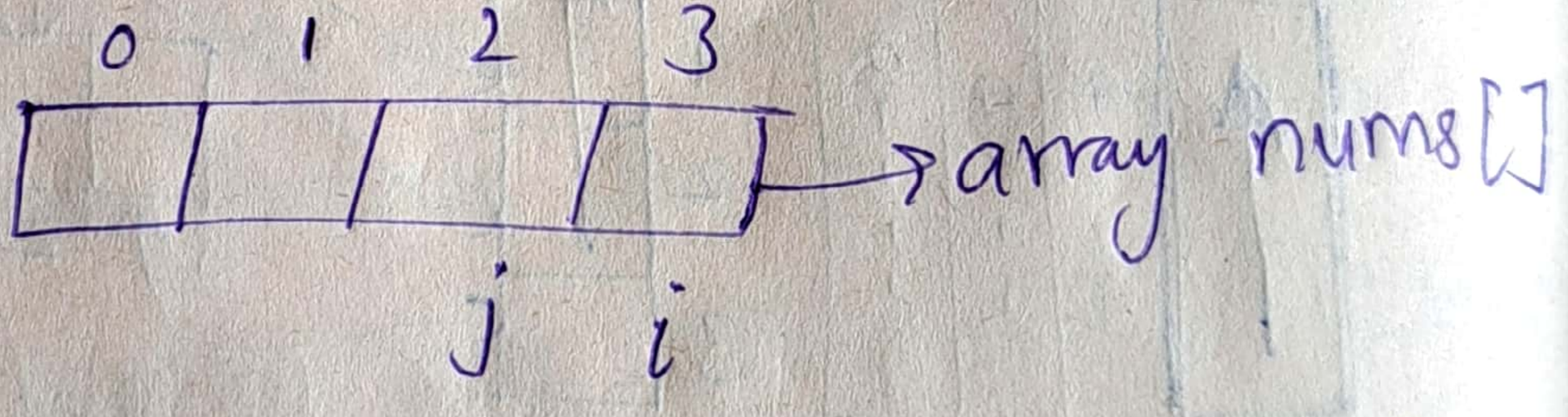
Similarly for the left() and right()

But why like this:-

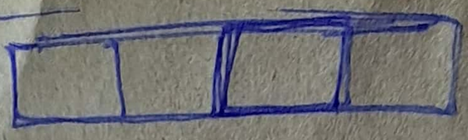
* simple answer is it improves the readability and reduces the no. of lines.
* Note: it will not enhances the time complexity

Swipe_right():-

* So, how does this work and how we are approaching to solve this
* basically iam taking two pointers i, j. they are going to start from i=3, j=2.



* if the nums[i] == 0 then we need to find a non zero element to swap it with
* So we are taking the zero from the right side of nums to left.



Start i at 3, j at 2

decrese j until a non-zero element occurs and j>0.

now compare i, j

if nums[i] == 0

then we need to swap nums[i] and nums[j]

if nums[i] > 0

then we need to compare i, j.

if they were equal then add them & nums[j] = 0.

now we performed nuceseary operation at i so we need to decrement i.

Now lets code it:

```
void swipe_right(int nums[]).
  { int i=3, j=2;
    while(j>=0 && nums[j]=0)
      { j--;
      }
    if(j<0) break;
```

```
if(nums[i]==0)
  {  swap(nums[i], nums[j]);
  }
else{  if(nums[i]==nums[j])
       {  nums[i]=2*nums[i];
          nums[j] = 0;
       }
     i--;
     if(i<=j)
       {  j=i-1;
       }
     }
  }
```

```
        }
        i--;
    }
  }
}
```

this is the code for ~~first~~ ~~non-zero~~ swiping.

---

hey, do you know this,

* we need to generate a value 2 or 4 in the empty cells.
* lets write a function to do that

```
void randomgenerate_block()
{ vector<pair<int,int>> emptycells;
  for(int i=0 to 4)
    { for(int j=0 to 4)
        { if(grid[i][j]==0)
            { emptycells.push_back({i,j});
  } }
```

```
if(!emptycells.empty())
  {
    int randomIndex = rand()%emptycells.size();
    int i= emptycells[randomIndex].first;
    int j =                    ].second;
    grid[i][j] = (rand()%2==0)?2:4;
  }
}
```

thats it

* we first need to know what are the empty locations available
* then we can generate a number either 2 or 4 it those empty cells.
* so create a pair vector to store those empty cells.
* if it is not empty then generate a random ~~index~~ index from the size of that Pair vector.
* then place either 2 or 4 in that cell

every time we ~~generate~~ made a move we need to find wether there is an empty space or not and are there any adjacent elements with same value or not.

```
bool is_game_over()
{  for (int i=0; i<4; i++)
     { for (int j=0; j<4; j++)
          { if(grid[i][j] ==0) return false;
            if(j<3 && grid[i][j]== grid[i][j+1])
                 return false;
            if(i<3 && grid[i][j]== grid[i+1][j])
                 return false;
          }
     }
return true;
}
```

Some times there may be no change in the grid for specific move, at that time we should not generate the new cell. to know that we need to keep track of Previous State.

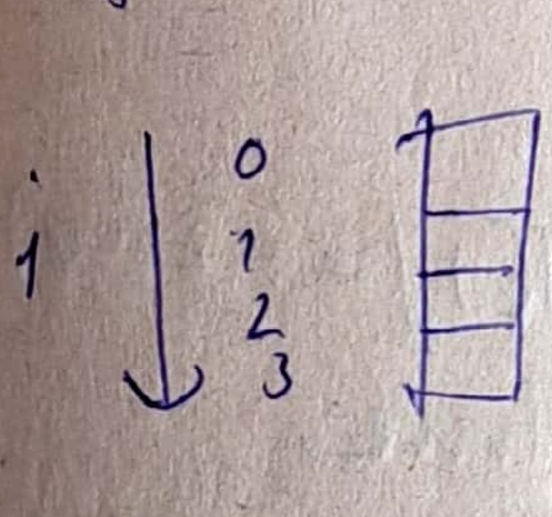
For that reason we are using a function known as altered()

```
bool altered()
{ for(int i=0 to 4)
  { for(int j=0 to 4)
    { if(temp[i][j] != grid[i][j])
        return true;
    }
  }
  return false;
}
```

now lets come to moving part:-

* in this part first we need to creat the array nums to pass it to swipe_right().
* each direction has a specific nums.

for up



here nums[3-j] = grid[j][i]

similarly for down

nums[j] = grid[j][i]

num[3-j] = grid[i][j] for left

nums[j] = grid[i][j] for right

* after creating nums the pass it to the swipe_right().

* then keep the nums array in the grid in the same way as taken.

* after performing every move we need to check whether the game_is_over() and is there any change with previous state.

* we also need to check if the game is won or not.

* to win the game there should be 2048 in the cell

```
bool is_game_won(){
    for(i=0 to 3)
    { for(j=0 to 3)
        { if(grid[i][j]==2048)
            return true;
        }
    }
return false;
}
```

now implementing move function.

```
bool move(int dir)
{ copyarray(); // Keeping the copy of
                  Previous state.
for(int i=0;i<4;i++)
  { int nums[4]={0};
    for(int j=0;j<4;j++)
      { switch(dir)
        {
          case 0: // UP
            nums[3-j] = grid[j][i];
            break;
          case 1: // down
            num[j] = grid[j][i];
            break;
          case 2:
            nums[3-j] = grid[i][j];
            break;
          case 3:
            nums[j] = grid[i][j];
            break;
        }
      }
    Swipe_right(nums);
for(int j=0; j<4; j++)
  { switch (dir){
      keep them as taken;
    }
  }
```

```
if(is_game_won()){
        iswon=true;
    }
if(!is_game_over() && altered())
{   randomgenerate_block();
    return true;
}
else if(is_gare_over())
    {   isended=true;
    }
    return false;
```

and this is it →

to know/see the total code visit my github.

" github.com/PRASAD0521 "

" github.com/PRASAD0521/2048-game "

made by Vara Prasad