02_Queimada

#Objetivo:

- Aprensentando a profundidade com um número como no vídeo da atividade.
- Conte a quantidade de árvores queimadas através do retorno da recursão. Não use nenhum tipo de variável global.
- Marque os locais por onde a recursão já voltou.

* O começo do codigo foi feito em sala de aula onde o professor deu algumas funções e explicou o que cada uma fazia, como a função embaralhar, na qual essa função retorna o vetor embaralhado as suas posições e foi ultilizado o swap, o getneib que significa pegar vizinhos, ele pega uma posição e retorna todas as posições ao seu lado. Como atividade extra o professor pediu pra implementar o contador e fazer o caminho da recursão, segue o codigo com isso feito!
Eu criei na função um inteiro chamado nivel e coloquei uma condição para ele fazer o caminho da recursão, em seguida la embaixo eu coloquei o contador pra receber a função queimar e retornar cont+1, onde ele irá mostrar quantas arvores foram queimadas.
* Foi um codigo bastante dificil de se fazer, na qual tive que assitir a video aula umas de muitas vezes para entender algumas coisas, mas aos poucos vou conseguindo aprender c++ e suas funções!
O tempo pra fazer essa atividade foi em media uns 7 dias mais ou menos!!

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <vector>

using namespace std;
const int nc = 60;
const int nl = 20;

struct Par{
    int l, c;
    //construtor
    Par(int l = 0, int c = 0){
        this->l = l;
        this->c = c;
    }
};

vector<Par> getNeib(Par p);
void mostrar(vector<string> &mat);
vector<Par> embaralhar(vector<Par> vet);
int queimar(vector<string> &mat, int l, int c,int nivel);

int main()
{
    srand(time(NULL));
    vector<string> mat(nl, string(nc, ' '));
    int narvores = nl * nc * 1.5;
   // insere varias arvores em lugares aleatorios
    //muitas serao superpostas
    for(int i = 0; i < narvores; i++){
        mat[rand() % nl][rand() % nc] = '#';
    }
```

```
    mostrar(mat);

    int total = queimar(mat, 0, 0, 0);
    cout << total << " arvores queimaram\n";

    return 0;
}

vector<Par> getNeib(Par p){
    vector<Par> v;

   v.push_back(Par(p.l, p.c + 1));
   v.push_back(Par(p.l, p.c - 1));
   v.push_back(Par(p.l + 1, p.c));
   v.push_back(Par(p.l - 1, p.c));
    return v;
}

void mostrar(vector<string> &mat){
    cout << string(50, '\n');
    for(string s : mat)
        cout << s << endl;
    getchar();
}

vector<Par> embaralhar(vector<Par> vet){
    for(int i = 0; i < (int) vet.size(); i++){
        int aleat = rand() % vet.size();
        std::swap(vet[i], vet[aleat]);
    }
    return vet;
}

int queimar(vector<string> &mat, int l, int c, int nivel){
    if(l < 0 || l >= nl)
        return 0;
    if(c < 0 || c >= nc)
        return 0;
    if(mat[l][c] != '#')
        return 0;
    if(nivel > 9)
        nivel = 0;

    mat[l][c] = '0' + nivel;

    int cont = 0;
    mostrar(mat);

    //for range
    for(Par p : embaralhar(getNeib(Par(l, c))))
        cont += queimar(mat, p.l, p.c, nivel + 1);

    return cont + 1;
}
```