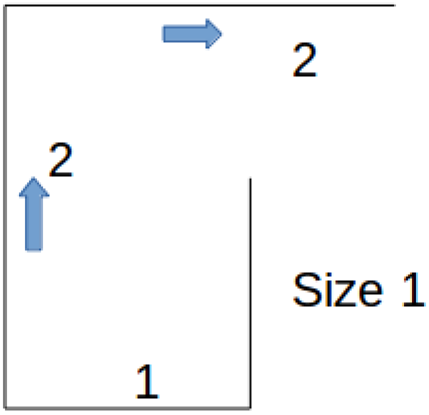
Introduction

The idea is to draw a Ulam spiral on the terminal link to Ulam Spiral definition :
https://en.wikipedia.org/wiki/Ulam_spiral

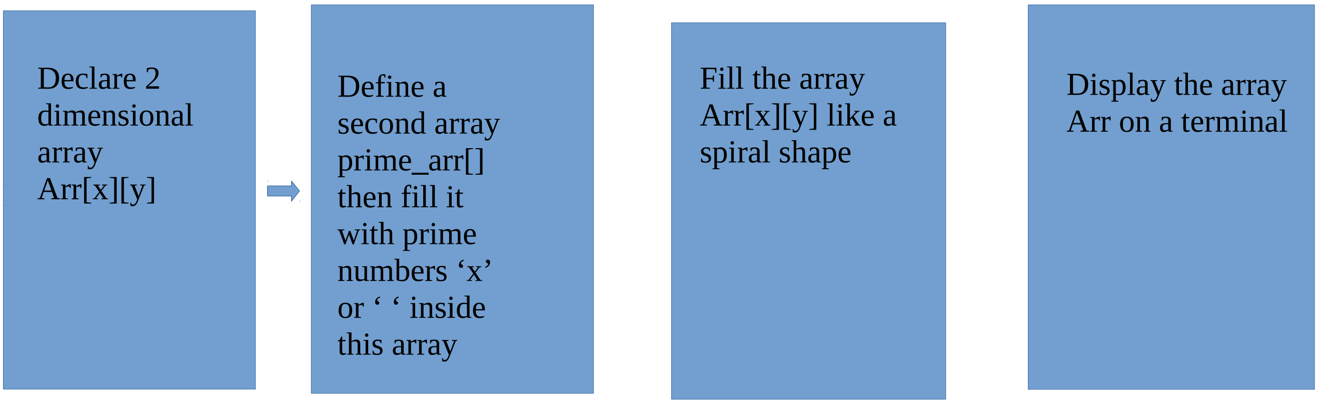


Principle of this piece
of Code:

Declare arrays and affects each element with a symbol
Print the array on the terminal iteratively and according to array order

Assign to array ‘ ‘ when the number is not a prime and ‘x’ when it is

Diagram



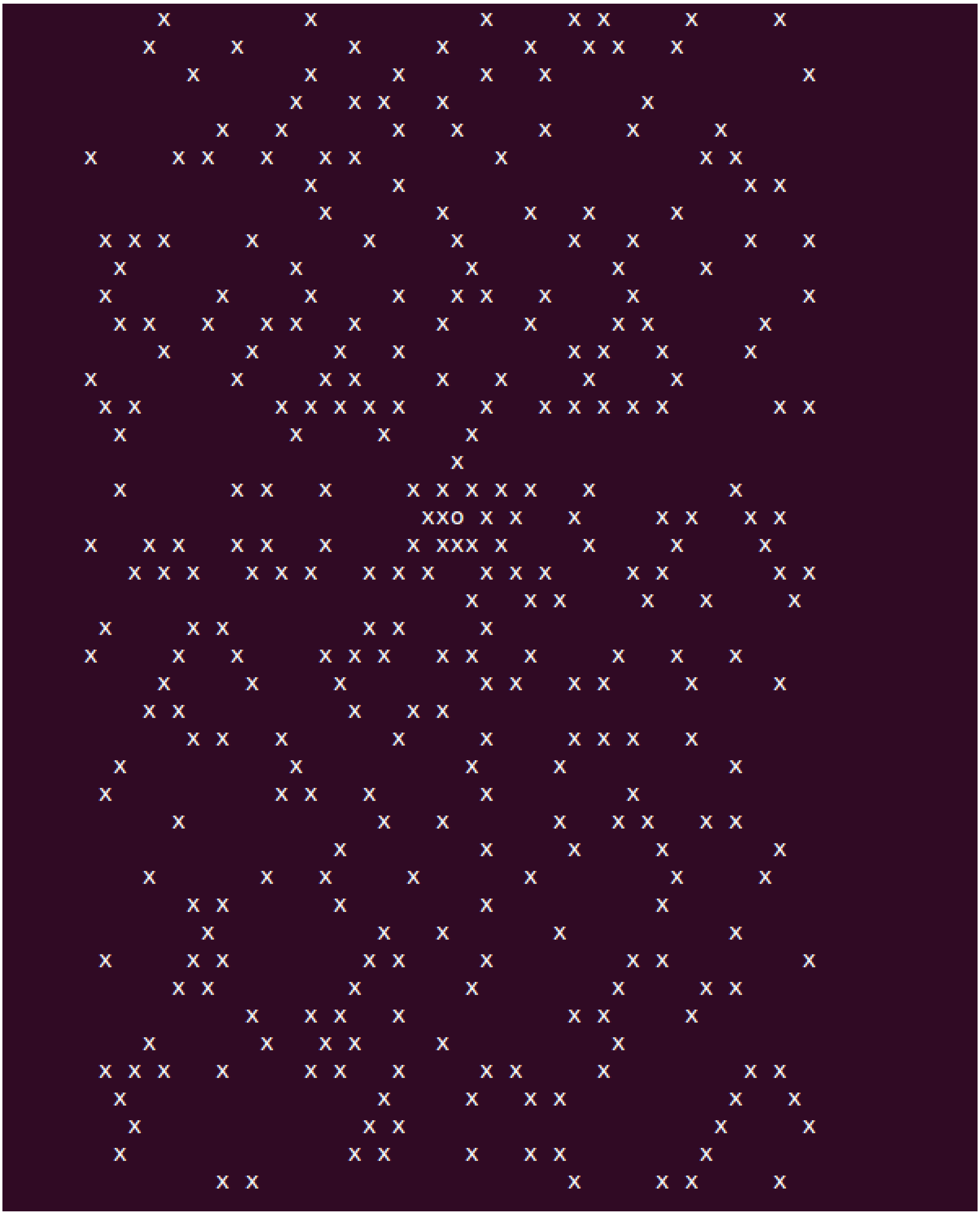
Declare 2 dimensional array Arr[x][y]

Define a second array prime_arr[] then fill it with prime numbers ‘x’ or ‘ ‘ inside this array

Fill the array Arr[x][y] like a spiral shape

Display the array Arr on a terminal

Screenshot



```
     X         X           X     X X     X     X
    X     X       X     X     X   X X   X
       X       X     X     X   X                 X
              X   X X   X             X
         X   X       X   X     X     X     X
X     X X   X   X X         X             X X
               X     X                       X X
                X       X     X   X     X
 X X X     X       X     X       X   X       X   X
  X           X           X         X     X
 X       X     X     X   X X   X     X           X
  X X   X   X X   X     X     X     X X       X
     X     X     X   X           X X   X     X
X         X     X X     X   X     X     X
 X X         X X X X X     X   X X X X X       X X
  X           X     X     X
                         X
  X       X X   X     X X X X X   X         X
                       XXO X X   X     X X   X X
X   X X   X X   X     X XXX X     X     X     X
   X X X   X X X   X X X   X X X     X X       X X
                          X   X X     X   X     X
 X     X X         X X     X
X     X   X     X X X   X X   X     X   X   X
     X     X     X         X X   X X     X     X
    X X           X   X X
       X X   X       X     X     X X X   X
  X           X           X     X           X
 X           X X   X       X         X
      X             X   X       X   X X   X X
                 X         X     X     X       X
    X       X   X     X       X         X     X
       X X       X         X           X
        X           X   X       X           X
 X     X X         X X     X         X X         X
      X X         X       X         X     X X
           X   X X   X           X X     X
    X       X   X X     X           X
 X X X   X     X X   X     X X     X         X X
  X                 X     X   X X           X   X
   X               X X                     X     X
  X               X X     X   X X         X
         X X                     X     X X     X
```

Program principles

Algorithm explanation :

Based on the ‘actualization’ of coordinate of a current cell, the programm at each loop knows these like Point(X,Y) then calculates where the new point will be placed point_two(X2,Y2) then fills the gap between these two cell locations iteratively.
2 different cases : up /down or Right/left and vertical or horizontal
Vertical ? Yes : Up or down ?

Horizontal ? Yes : Left or Right ?

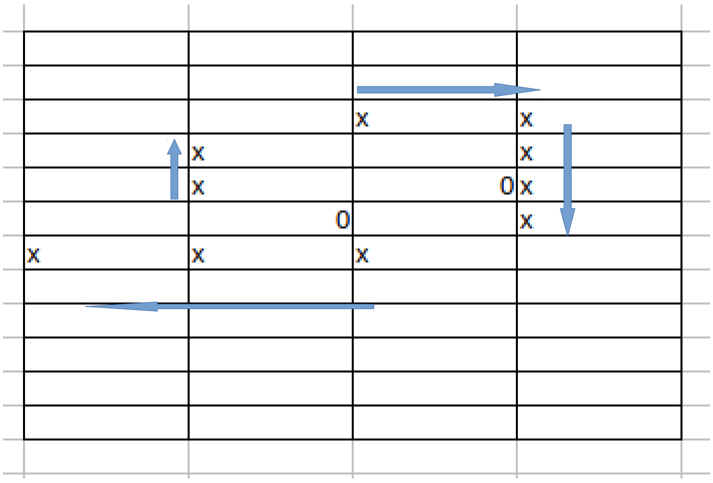
Algo parameters :

The % operator on parity and the sign of myPow(x,n) makes it alternate around.

The number of elements to be written is increased after each looping if n counts the nbr of turns then n symbol are assigned at each n symbols (‘ ‘ or ‘x’) are assigned at each looping.

This Algo starts with n=2 then the initialization step must be hard coded; by choice

schematics one



Display of the array

Functions defined and libraries

How primes are calculated : % and iteratively with an int 64 bit-Datatype

<stdio.h> for printf

<stdbool.h> for TRUE and FALSE datatypes

Source Code

```
#include<stdio.h>

#include<stdbool.h>

int myPow(int a, int b);

bool toCheckifPrime(int a);
```

```
}


int main(){


char tab[100][100]; //1[] :up Or down 2[] : Right or LEFT in tab[][]

char prime_array[3000];


int X=49;

int Y=49;

int v=2;

int f=2;

int ny=0;//counter inside prime_array array


/*Initialize arays*/


for(int g=0;g<3000;g++){

        if(toCheckifPrime(g)==true){

                prime_array[g]='x';

        }

        else prime_array[g]=' ';


}


for(int g=0;g<100;g++){

        for(int u=0;u<100;u++){

                tab[g][u]=' ';

        }


}

//Initialization of a spiral

tab[50][50]='o';

tab[49][49]='x';
```

```
/*Fill the tab draw the spiraL*/


for(int g=2;g<100;g++){
    if(g%2==0){


    if(myPow(-1,v)==-1){
        for(int i=X-1;i>=X+myPow(-1,v)*v;i--){
                tab[i][Y]=prime_array[ny];
                ny++;
            }
        }
    if(myPow(-1,v)==1){
        for(int h=X+1;h<=X+myPow(-1,v)*v;h++){
                tab[h][Y]=prime_array[ny];
                ny++;
            }
        }
            X=X+myPow(-1,v)*v;
            v++;



                    }


    else{


            if(myPow(-1,f)==-1){
        for(int r=Y-1;r>=Y+myPow(-1,f)*f;r--){
                tab[X][r]=prime_array[ny];
                ny++;
            }
        }
        if(myPow(-1,f)==1){
            for(int w=Y+1;w<=Y+myPow(-1,f)*f;w++){
                tab[X][w]=prime_array[ny];
                ny++;
            }
            }
            Y=Y+myPow(-1,f)*f;
            f++;


                    }

}



/*Display the tab*/
for(int l=0;l<100;l++){
   printf("\n");
    for(int u=0;u<100;u++){
        printf("%c", tab[l][u]);


    }
}

 return 0;

}
```

```
/*Function returns True is the integer input is a prime number, false otherwise integer >=0*/
bool toCheckifPrime(int a){
	bool outPut=true;
	int inter=0;
	for(int i=2;i<a;i++){
		inter=a%i;
		if(inter==0){
			outPut=false;
			break;
		}

	}
	return outPut;
}


int myPow(int a, int b){

	int number=1;
	for(int k=0;k<b;k++){
		number=number*a;
	}
	return number;
}
```