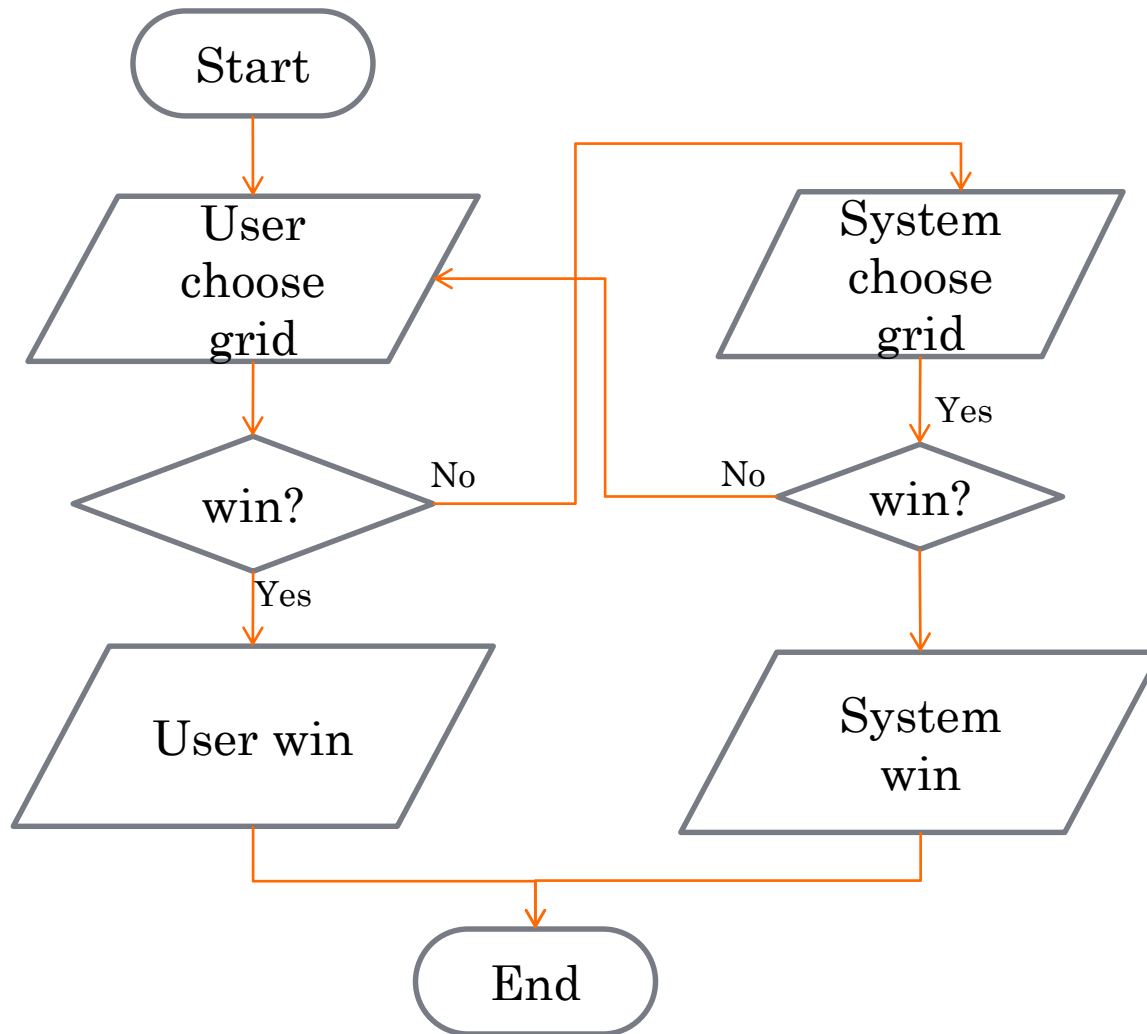


INTRODUCTION TO NOUGHTS AND CROSSES

- The object of Noughts and Crosses is to get a winning line of three Noughts or three Crosses in either a horizontal, vertical or diagonal row.



FLOW CHART

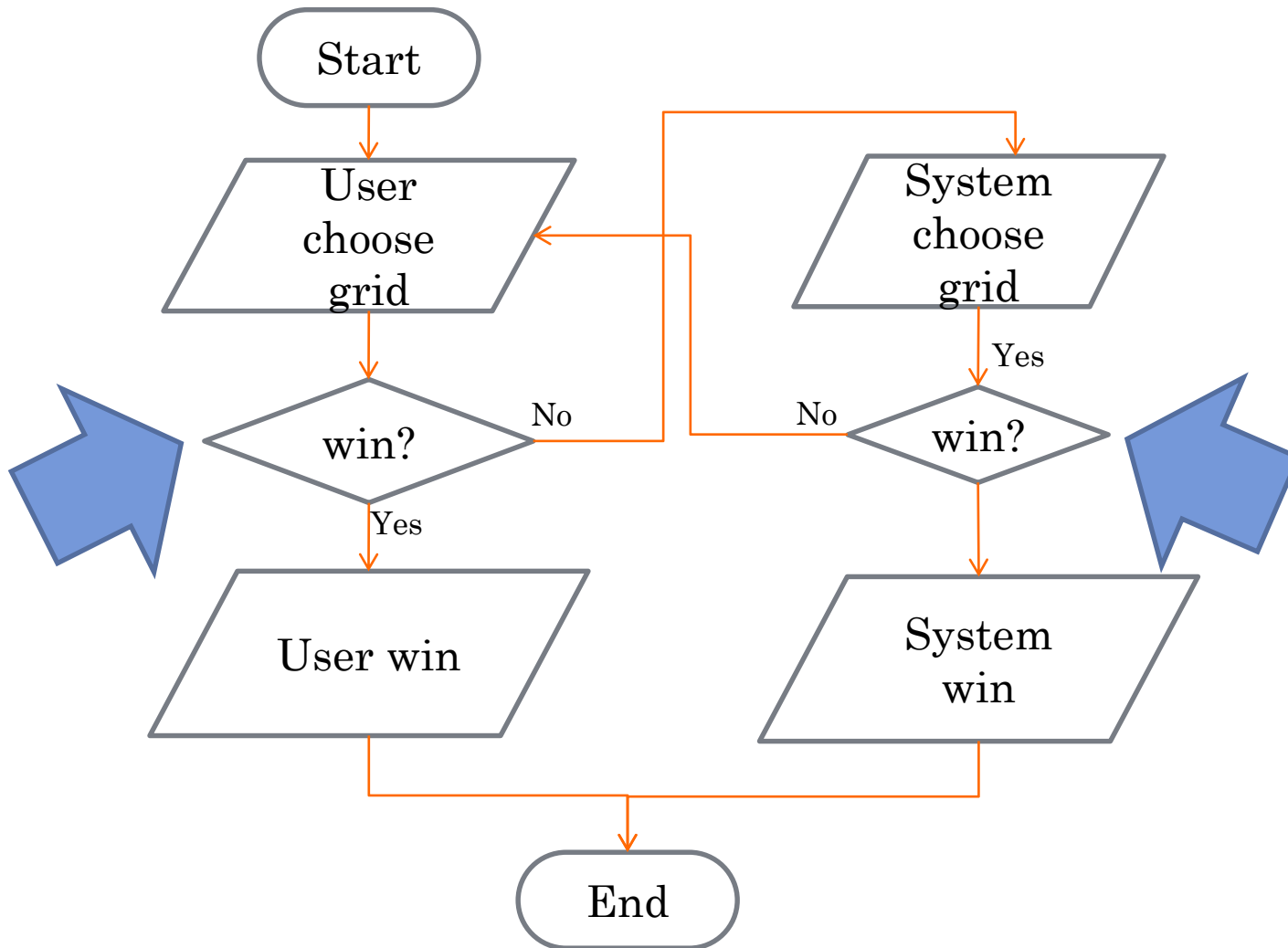


PARAMETERS PASSING

```
#include<iostream>
using namespace std;
int test(int a[], int b)
{
    return a[0]+a[1]+a[2]+b;
}
void main()
{
    int array[3]={0,0,0};
    int b=3;
    int catch=test(array,b);
    cout<<catch;
    system("PAUSE");
}
```

Test function

FLOW CHART



LINE CHECKING

O	X	
O	O	
X		X

grid[0]=0	grid[1]=1	grid[2]=2
grid[3]=0	grid[4]=0	grid[5]=2
grid[6]=1	grid[7]=2	grid[8]=1

```
int check_win(int grid[])//X win(return 1)or O win(return 0) or not yet(return 2)
```

```
{
```

```
//the size of grid is 9 (from 0 to 8)
//if value of grid[i] is 2, means that it draw nothing
//if value of grid[i] is 0, means that it draw O
//if value of grid[i] is 1, means that it draw X
//now we only have grid[]
//please check is any line has three 0 or three X
//three 0 in one line, please return 0
//three 1 in one line, please return 1
//otherwise return 2(means no one win yet)
```

```
}
```

LINE CHECKING

O	X	
O	O	
X		X

grid[0]=0	grid[1]=1	grid[2]=2
grid[3]=0	grid[4]=0	grid[5]=2
grid[6]=1	grid[7]=2	grid[8]=1

```
int check_win(int grid[])//X win(return 1)or O win(return 0) or not yet(return 2)
{
    for(int i=0;i<3;i++)
    {
        if(grid[i]==grid[i+3] && grid[i]==grid[i+6] && grid[i]!=2)
            return grid[i];
        if(grid[i*3]==grid[i*3+1] && grid[i*3]==grid[i*3+2] && grid[i]!=2)
            return grid[i*3];
    }
    if(grid[0]==grid[4]&& grid[0]==grid[8] && grid[0]!=2)
        return grid[0];
    if(grid[2]==grid[4]&& grid[6]==grid[4] && grid[2]!=2)
        return grid[2];
    return 2;
}
```



LINE CHECKING

0	1	2
3	4	5
6	7	8

```
int check_win(int grid[]) //X win(return 1) or O win(return 0) or not yet(return 2)
{
    for(int i=0; i<3; i++)
    {
        if(grid[i]==grid[i+3] && grid[i]==grid[i+6] && grid[i]!=2)
            return grid[i];
        if(grid[i*3]==grid[i*3+1] && grid[i*3]==grid[i*3+2] && grid[i]!=2)
            return grid[i*3];
    }
    if(grid[0]==grid[4] && grid[0]==grid[8] && grid[0]!=2)
        return grid[0];
    if(grid[2]==grid[4] && grid[6]==grid[4] && grid[2]!=2)
        return grid[2];
    return 2;
}
```



LINE CHECKING

i=0

	0	1	2
	3	4	5
	6	7	8

```
int check_win(int grid[]) //X win(return 1) or O win(return 0) or not yet(return 2)
```

```
{
```

```
for(int i=0;i<3;i++)
```

```
{
```

```
if(grid[i]==grid[i+3] && grid[i]==grid[i+6] && grid[i]!=2)
```

```
return grid[i];
```

```
if(grid[i*3]==grid[i*3+1] && grid[i*3]==grid[i*3+2] && grid[i]!=2)
```

```
return grid[i*3];
```

```
}
```

```
if(grid[0]==grid[4]&& grid[0]==grid[8] && grid[0]!=2)
```

```
return grid[0];
```

```
if(grid[2]==grid[4]&& grid[6]==grid[4] && grid[2]!=2)
```

```
return grid[2];
```

```
return 2;
```

```
}
```



LINE CHECKING

i=1

0	1	2
3	4	5
6	7	8

```
int check_win(int grid[]) //X win(return 1) or O win(return 0) or not yet(return 2)
```

```
{
```

```
for(int i=0;i<3;i++)
```

```
{
```

```
if(grid[i]==grid[i+3] && grid[i]==grid[i+6] && grid[i]!=2)
```

```
return grid[i];
```

```
if(grid[i*3]==grid[i*3+1] && grid[i*3]==grid[i*3+2] && grid[i]!=2)
```

```
return grid[i*3];
```

```
}
```

```
if(grid[0]==grid[4]&& grid[0]==grid[8] && grid[0]!=2)
```

```
return grid[0];
```

```
if(grid[2]==grid[4]&& grid[6]==grid[4] && grid[2]!=2)
```

```
return grid[2];
```

```
return 2;
```

```
}
```



LINE CHECKING

0	1	2
3	4	5
6	7	8

i=2

```
int check_win(int grid[]) //X win(return 1) or O win(return 0) or not yet(return 2)
```

```
{
```

```
for(int i=0;i<3;i++)
```

```
{
```

```
if(grid[i]==grid[i+3] && grid[i]==grid[i+6] && grid[i]!=2)
```

```
return grid[i];
```

```
if(grid[i*3]==grid[i*3+1] && grid[i*3]==grid[i*3+2] && grid[i]!=2)
```

```
return grid[i*3];
```

```
}
```

```
if(grid[0]==grid[4]&& grid[0]==grid[8] && grid[0]!=2)
```

```
return grid[0];
```

```
if(grid[2]==grid[4]&& grid[6]==grid[4] && grid[2]!=2)
```

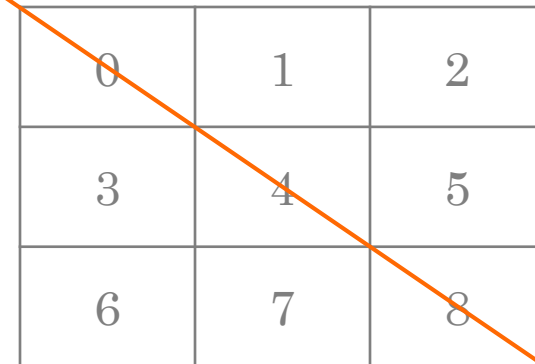
```
return grid[2];
```

```
return 2;
```

```
}
```



LINE CHECKING



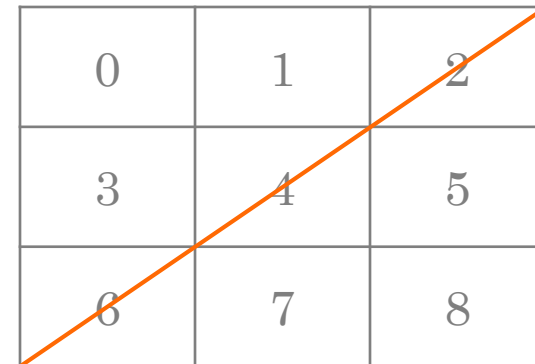
0	1	2
3	4	5
6	7	8

```
int check_win(int grid[])//X win(return 1)or O win(return 0) or not yet(return 2)
{
    for(int i=0;i<3;i++)
    {
        if(grid[i]==grid[i+3] && grid[i]==grid[i+6] && grid[i]!=2)
            return grid[i];
        if(grid[i*3]==grid[i*3+1] && grid[i*3]==grid[i*3+2] && grid[i]!=2)
            return grid[i*3];
    }
    if(grid[0]==grid[4]&& grid[0]==grid[8] && grid[0]!=2)
        return grid[0];
    if(grid[2]==grid[4]&& grid[6]==grid[4] && grid[2]!=2)
        return grid[2];
    return 2;
}
```



LINE CHECKING

0	1	2
3	4	5
6	7	8



```
int check_win(int grid[])//X win(return 1)or O win(return 0) or not yet(return 2)
{
    for(int i=0;i<3;i++)
    {
        if(grid[i]==grid[i+3] && grid[i]==grid[i+6] && grid[i]!=2)
            return grid[i];
        if(grid[i*3]==grid[i*3+1] && grid[i*3]==grid[i*3+2] && grid[i]!=2)
            return grid[i*3];
    }
    if(grid[0]==grid[4]&& grid[0]==grid[8] && grid[0]!=2)
        return grid[0];
    if(grid[2]==grid[4]&& grid[6]==grid[4] && grid[2]!=2)
        return grid[2];
    return 2;
}
```



