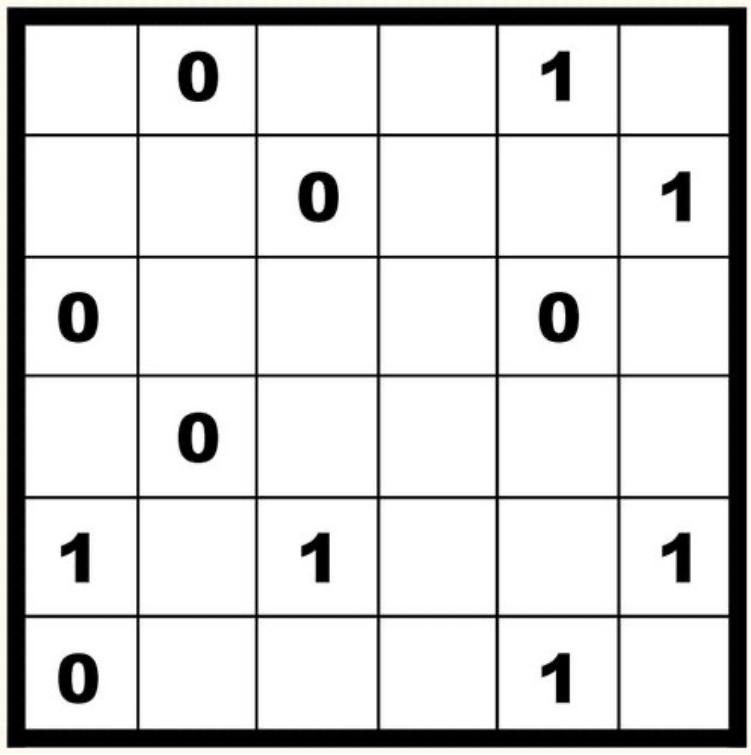
Cyril Nakhla
Eyfeline Tala
INT 4

Programming in C - Takuzu project



Year 2021-2022 – Professors Kaïs Klaï & Hanen Hochi

Table of content:

I – Introduction :

The “Takuzu” is a project based on the creation of a game, also called “Binero”, whose goal is to fill the cells with “1” and “0” of a square matrix by respecting different rules such as :

- We can't have 3 same numbers side by side
- Each row and each column must contain the same number of "0" as "1”
- Each row and each column must be unique. There cannot be 2 similar rows or 2 similar columns

The main objectives of this project were to reinforce our learning by allowing us to practice more, it also allowed us to face several constraints such as semantic and syntactic errors and the fact of making real what we intended to do.

II – Functionality and innovation :

1) requested functions

- Part 1: Let a player solve a grid

✓ Play & Solve a grid (4x4, 8x8 & 16x16)
✓ Player’s life
✓ Valid & Correct move
✓ Display a clue
✓ Enter manually a mask / Generate automatically a mask & use it

- Part 2: Automatically solve a grid

✓ To complete

- Part 3: Generate a grid

✓ Display all valid rows
✓ Generate a grid of size : 4x4, 8x8 and 16x16

2) Extra functions

- Creation of a complete menu, governing the whole code (see result presentation)
- Creation of an interactive interface which talk with you
- Creation of a functionality that we used to allow the user to quit the program when he wants.
- Creation of *a functionality* which allows us to secure the inputs
- The possibility to play with a mask that we entered
- The possibility to play with a generated mask
- The possibility to play a random grid 4x4, 8x8 & 16x16
- The possibility to display all line possible for a 16x16 grid

III – Technical aspect :

1) Essential algorithms

A – Solve a grid :

The most difficult part to implement at the level of matrices was the **creation**. To create this part we had to:

- For the mask :

- Check if the input is correct
- Store in a matrice the 1 and 0 of the mask
- Check if the mask is possible to play or not *autopossible()*
- Create a third matrice to create the one playable *print_pad_mask()*

- About the playable part of a grid :

- Check if the input is correct
- Check each rule of TAKUZU Game to see if a movement is valid *check_valif_off()*
- Compare each move with the final solution to see if a move is correct
- Fill automatically a grid *auto_fill()*
- Display a grid *print_pad()*
- Make that the initial mask can't be modify during the game

```
int check_valid_off(int** L,int value, int i, int j, int check, int size){
    if ((value==0) && (L[i][size]>=size/2)){
        return 4;
    }
    else if ((value==0) && (L[size][j]>=size/2)){
        return 5;
    }
    else if ((value==1) && (L[i][size+1]>=size/2)){
        return 2;
    }
    else if ((value==1) && (L[size+1][j]>=size/2)){
        return 3;
    }
    if ((i<size && i>0) && (j<size && j>0) && (value == 0) && (L[i-1][j]==0 && L[i+1][j]==0) && (L[i][j-1]==0 && L[i][j+1]==0)){
        return 1;
    }
    else if ((i<size && i>0) && (j<size && j>0) && (value == 1) && (L[i-1][j]==1 && L[i+1][j]==1) && (L[i][j-1]==1 && L[i][j+1]==1)){
        return 0;
    }
    if ((i == 0 || i == size) && (value == 0) && (L[i][j-1]==0 && L[i][j+1]==0)){
        return 1;
    }
    else if ((i == 1 || i == size) && (value == 1) && (L[i][j-1]==1 && L[i][j+1]==1)){
        return 0;
    }
    if ((j == 0 || j == size) && (value == 0) && (L[i][j-1]==0 && L[i][j+1]==0)){
        return 1;
    }
    else if ((j == 1 || j == size) && (value == 1) && (L[i+1][j]==1 && L[i-1][j]==1)){
        return 0;
    }
```

Little part of the check_valif_off()'s function

B – Generation of a grid

Based on the grid creation part, the tasks were long to perform. It was necessary to translate numbers into binary before selecting only the rows respecting the rules of TAKUZU (same number of 1's as 0's and not more than 2 times the same number next to each other). After that, we had to connect them to form a complete matrix respecting again the rules of TAKUZU.

- `display_all_valid_rows(int size)`
- `generate_grid(int size, bool* created, bool automatic)`

2) Organization of the code :

To start the project, we decided to organize our program in **several functions**, on the one hand functions related to the menu, to avoid errors that could be encountered by the user and on the other hand more aesthetic features allowing the console to return more readable information. On a technical level, this allowed us to avoid repetitions and to make a more efficient program.

3) Technical constraints :

- Make sure that the console can always return something for any input
  - Creation of a menu + use of while loops
- Link all the functions to the different menus
  - Use well-defined functions in a precise order at precise places
- Pay attention to the choice of the size of the **matrices**
  - Create functions that can work for any size to avoid repetition
- Try a mask on a matrice to see if it is possible
  - Create & integration of a matrice to see if this mask is playable
- Check the choices of the player
  - Loose a life if his movement is incorrect
- Avoid the modification of a mask
  - Lock the value of a mask with its coordinate

III/ Result presentation

1) Data used

```
void Menu_principal(int value){
    int choice;
    if (value == 0) {
        printf("================================================================\n\n"
               "   =======      ==        ==     ==  ==    ==   ======   ==    ==\n"
               "      =       ==  ==      ==    ==   ==    ==      ==    ==    ==\n"
               "      =      == == ==     ==   ==    ==    ==     ==    ==    ==\n"
               "      =     ==       ==    ==     ==     ====     ======     ====\n\n"
               "================================================================\n\n");
        printf("Hi !\nWelcome in our new game : The TAKUZU !\n");
        printf("\nThere what you can do in my world ! Just ask and admire :p\n  1. Solve a grid\n  2. Automatically solve a grid\n  3. Generate a grid\n\n  -> "
        scanf("%d", &choice);
    }
    else if (value == 1){
        printf("Here we go again in our new game : The TAKUZU !\nI'm surprise, I thought you might be tired of it");
        printf("\nThere what you can do in my world ! Just ask and admire :p\n  1. Solve a grid\n  2. Automatically solve a grid\n  3. Generate a grid\n\n  -> "
        scanf("%d", &choice);
    }
    if (choice == 1){
        Menu_second_1();
    }
    else if (choice == 2){
        Menu_second_2();
    }
}
```

```
void Menu_second_1(){
    int choice, size, **grid;
    int ** grid4 = array();
    int ** grid8 = array();
    printf("\n\nSo you wanna play ?! Let's play !\n");

    printf("\nBut firstly, enter the size in which you want to play please :\n  1. 4x4\n  2. 8x8\n\n   -> ");
    scanf("%d", &choice);
    if (choice == 1){
        size = 4;
        grid = grid4;
    }
    else{
        size = 8;
        grid = grid8;
    }
    printf("\n\nThere what you can do ! Just ask and admire :p\n  1. Enter a mask manually\n  2. Automatically generate a mask\n  3. Play\n\n  -> "
    scanf("%d", &choice);

    if (choice==1){
        Menu_enter_mask(grid, size);
    }
    else if (choice == 2){
        Menu_auto_mask(grid, size);
    }
    else if (choice == 3){
        play_game_new(grid, size);
    }
}
```

```
void Menu_second_2(){
    int choice, size, **grid, **mask;
    int ** grid4 = array();
    int ** grid8 = array();
    printf("\n\nWait ! What ?? You're playing on my game and you want to me to work at your place ??\nWhat a world !\n");
    printf("\nPff, enter the size in which you want to play please :\n  1. 4x4\n  2. 8x8\n\n   -> ");
    scanf("%d", &choice);
    if (choice == 1){
        size = 4;
        grid = grid4;
    }
    else{
        size = 8;
        grid = grid8;
    }
    mask = Create_mask(size, grid);
    printf("\n\nThere is your matrice chef ! I will work on it right now !\n\n");
    int** grid_game = print_pad_mask(mask, grid, size);
    auto_fill(grid_game, size);
    printf("\nIT'S COMPLETE !!\n\n");
    display_pad(grid_game, size);
    printf("\n\nYour Game is finish, I hope you took some pleasure by playing.\nTo return to main menu, enter anything pleeeeaaase\n  -> ");
    scanf("%d", &choice);
    Menu_principal(1);
}
```

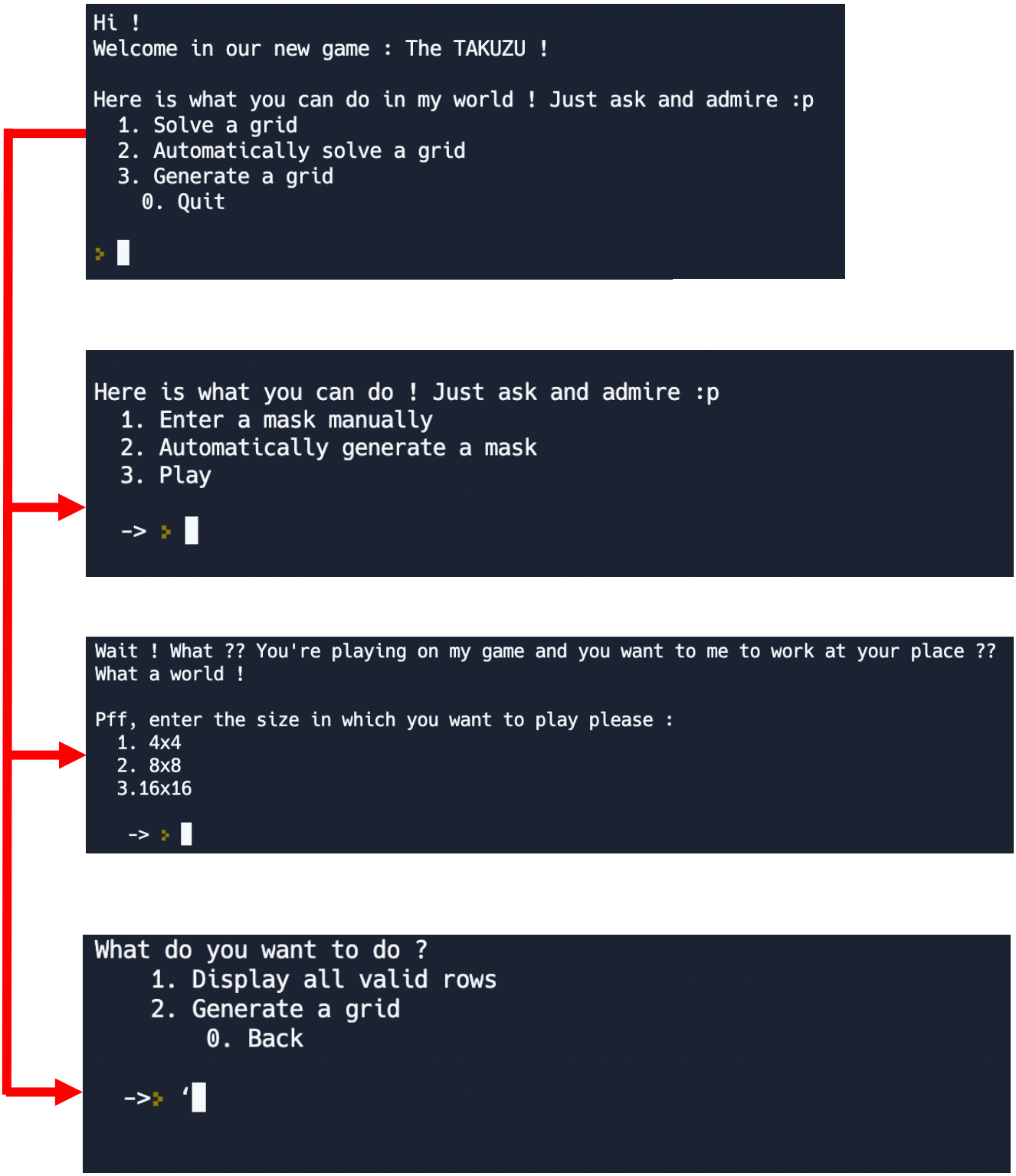
```c
bool menu_generation(int**full_grid)
{
    int input, size;
    bool created;
    printf("\n======= Generation =======\n");
    printf("What do you want to do ?\n");
    printf("    1. Display all valid rows\n");
    printf("    2. Generate a grid\n");
    printf("        0. Back\n\n  ->");
    other_entry_check(&input, 0, 2);
    if(input)
        Get_size(&size);
    if(input == 1)
    {
        display_all_valid_rows(size);
    }
    else if(input == 2)
    {
        full_grid = generate_grid(size, &created, false);
        return created;
    }
}
```

```
1 0 1 1 0 1 1 0 1 0 1 0 0 1 0 0
0 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1
0 0 1 0 1 1 0 1 0 1 0 1 1 0 1 0
1 1 0 1 0 1 1 0 1 0 0 1 0 0 1 0
0 0 1 0 1 0 0 1 0 1 1 0 1 1 0 1
1 0 1 0 0 1 0 1 0 1 1 0 0 1 0 1
0 1 0 1 1 0 1 0 1 0 0 1 1 0 1 0
0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 1
1 0 1 0 1 1 0 1 0 0 1 0 0 1 1 0
0 1 1 0 1 1 0 0 1 0 1 0 0 1 1 0
1 0 0 1 0 0 1 0 1 1 0 1 1 0 0 1
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 1 0 0 1 0 1 0 1 0
0 0 1 0 1 0 1 0 1 1 0 1 0 0 1 1
1 1 0 1 0 1 0 0 1 1 0 1 0 1 0 0
1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 1
```

```
1 1 0 1 0 1 0 1 1 0 0 1 0 0 1 0
1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0
0 1 1 0 0 1 1 0 0 1 0 1 1 0 0 1
1 0 0 1 1 0 1 0 0 1 0 1 0 1 0 1
1 0 0 1 0 1 0 1 1 0 1 0 0 1 1 0
0 1 1 0 1 0 0 1 1 0 1 0 1 0 0 1
1 0 1 0 0 1 1 0 0 1 0 1 0 0 1 1
0 1 0 1 0 0 1 1 0 1 1 0 1 1 0 0
0 1 0 0 1 1 0 0 1 0 1 1 0 1 0 1
1 0 1 0 1 0 0 1 1 0 0 1 1 0 1 0
1 0 1 1 0 1 1 0 0 1 0 0 1 1 0 0
0 1 0 1 0 1 1 0 0 1 1 0 0 1 0 1
0 0 1 0 1 0 0 1 1 0 0 1 1 0 1 1
1 0 1 0 1 0 0 1 0 1 0 1 1 0 1 0
0 1 0 1 0 1 1 0 1 0 1 0 0 1 0 1
0 1 1 0 1 0 1 0 0 1 1 0 1 0 1 0
```

Here are two 16x16 grids created using the program

2) Display on the console



PS: Some functionnallity like *auto_fill()* are displayed progressively thanks to "getch()" used in some functions, we found it more pleasant for the user.

IV – Conclusion

To conclude, I would say that this experience taught us more at a technical level, whether it is for the functions, the use of matrices but also at the level of the organization of the work, to communicate well to know the progress of our mate, to know if functions were already created...

Finally, I would say that this project was for us a new step in the world of computer science as such, it allowed us to experiment real constraints both in terms of time limits and skills.

We take only positive things out of it !