Introduction to Programming

— Practical #3 —

1 Arrays

In this practical, we will program a game of Mastermind, where the player has to guess a combination generated randomly by the computer. The user has a predetermined number of trials. For each compbination trial, the computer answers with two hints : the number of correctly placed pieces and the number of pieces of the right color but at a wrong position.

1.1 Text Mastermind

1. *Get the initial code :*
   Download the archive `Tp3_Initial.zip` from the course webpage, decompress it and open the project. Study the project `MasterMind`. Input your name in the placeholder of file `mastermind.cpp`.
2. *Represent a combination :*
   We will take here a combination of 5 pieces of 4 different colors. The color of a piece will be represented by an integer in the range from 0 to 3. For a combination of 5 pieces, we will therefore use an array of 5 integers.

   ```
   int combin[5]; // array of 5 integers
   ```

3. *Display a combination :*
   Code a function displaying a combination on the screen. The simplest manner it to display the different numbers of the combinations on the same line one after the other.
4. *Generate a random combination :*
   At the start of the game, the computer must generate randomly a combination for the user to guess. We will use for that the functions declared in file cstdlib, notably the function rand() allowing to generate a random number between 0 and RAND_MAX (a constant, which may vary depending on the compiler). To get a number between 0 and n, we can proceed like this :

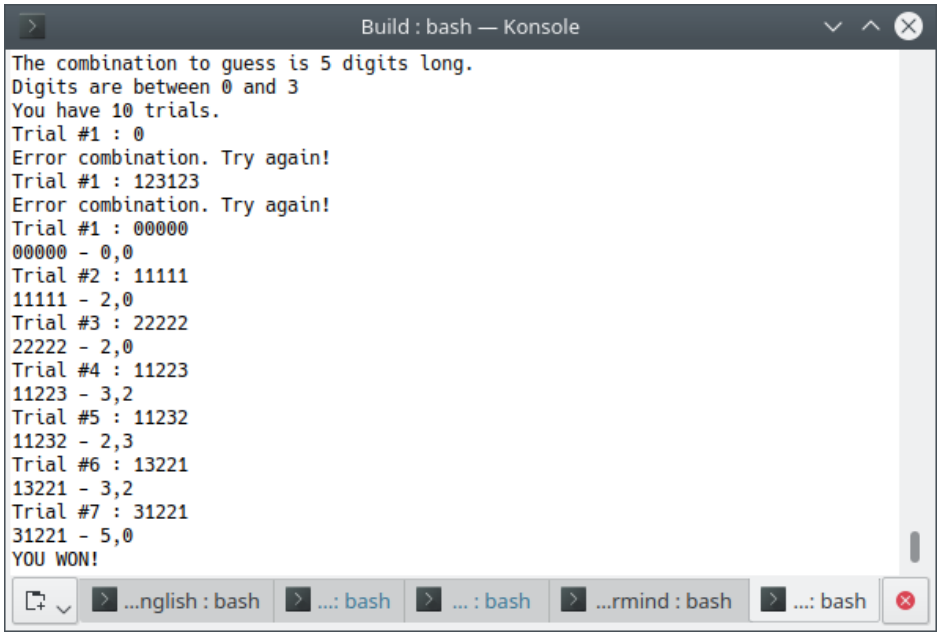
   ```
   int x = rand()%n;
   ```

   So that the sequence of generated numbers is not the same at each run of the program, it is necessary to initialize the generator with a variable seed. The simplest way is to use the current time. The function `time()` declared in file `ctime` gives access to it.
   Finally, our function allows generating a combination :

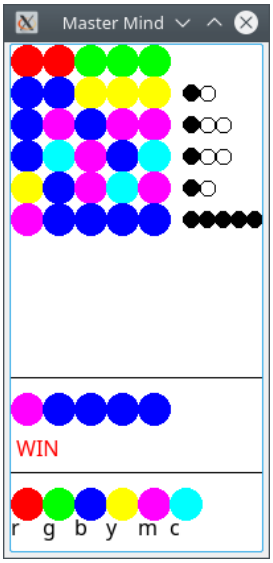
   ```
   #include <cstdlib>
   #include <ctime>
   using namespace std;

   void generateCombination(int combin[5]) {
       for(int i=0; i<5; ++i)
           combin[i] = rand()%4;    // calls to generator
   }
       srand((unsigned int)time(0)); // initialization
       generateCombination(combin);
   ```

5. *Change the game complexity :*
   You will soon become experts in Mastermind and will want to increase the difficulty. It is enough to extend the length of the combination, or change the number of possible colors. It is quite easy if you thought of using a global constant for each quantity. If not, it is not too late to remedy it. Define for example :



Mastermind in console...



Mastermind with graphics...

FIGURE 1 – The two forms of our game.

```
const int nbcases = 5; // length of combination
const int nbcol = 4;   // number of colors
```

Review the code you have written using these constants. It is very important to store constant parameters in variables, it saves a lot of effort when we need to change them.

6. *Input combination from the keyboard :*
The following function, that we will admit for now, get a *character string* (type string) from the keyboard and fills the array combi[] with the nbcases first characters of the string.

```
void getCombination(int trial, int combi[nbcases]) {
  cout << "Trial# " << trial << " : ";
  string s;
  cin >> s;
  for(int i=0; i<nbcases; i++)
     combi[i]=s[i]-'0';
}
```

Beware, the `cin` does not work from the integrated terminal in Qt Creator (that is why the window is called Application *Output*, it does not support Input), we need to use an external terminal. Go into mode Projects, section Run and select the button "Run in Terminal".

In the framework of our Mastermind, we are going to modify this function so that it controls the input string is of the correct size and that its digits are beween 0 and nbcol−1. The trial will be asked again until a valid combination is submitted. We will use the function s.size() that returns the size of a string `s` (the syntax of this function call [1] will be explained much later in the course...)

A small explanation on the line s[i]−'0'. Characters (of type `char`) are actually numeric values, their ASCII code. '0' gives the ASCII code of character zero. We can check it writing for example

```
cout << (int)'0' << endl;
```

(We need to cast to an integer, otherwise we ask to display a `char`, and then the computer prints the character associated to the ASCII code, 0!) This values is 48. It happens that digits from 0 to 9 have consecutive ASCII codes : '0'=48, '1'=49, '9'=57. The type string of variable s behaves like an array of char. Hence, s[i]−'0' is the integer 0 if s[i] is character 0, '1'-'0'=1 if it is the character 1, etc.

1. You could have expected something like size(s).

7. *Combination analysis :*
   We should now program a function comparing a given combination to the one to guess. It should return two values : the number of pieces of correct color at correct position, and, for the remaining pieces, the number of correct color but at a wrong position.
   For example, if the secret combination is 02113 :
   00000 : 1 piece well placed (0xxxx), 0 piece wrongly placed (xxxxx)
   20000 : 0 well placed (xxxxx), 2 wrongly placed (20xxx)
   13133 : 2 well placed (xx1x3), 1 wrongly placed (1xxxx)
   13113 : 3 well placed (xx113), 0 wrongly placed (xxxxx)
   12113 : 4 well placed (x2113), 0 wrongly placed (xxxxx)
   ...
   To begin and to be able to test right away the game, program first a function returning only the number of well placed pieces.
8. *Game loop :* We have now at disposal all the necessary fundation, [2] we just have to assemble them to have a mastermind game. Think also to add the detection of victory (all pieces are well placed), and of failure (number of trials have been reached, in which case the solution must be printed).
9. *Complete version :* Complete the function of combination comparison so that it returns also the number of wrongly placed pieces.

1.2 Mastermind in graphics mode

The mastermind game we built has a rudimentary user interface. We will fix it by adding a graphical user interface (GUI).

1. *Study of initial project :*
   Switch to project `MastermindGraphique` [3]
   Graphic functions are already defined. They work following a principle of division of the window in lines. The function :

   ```
   void displayCombination(int n, int combi[nbcases]);
   ```

   prints the combination *combi* at line `n`. At the start, we leave at the top of the window as many free lines as the number of trials in order to display the game. We display at the bottom a mini user guide that sums up the correspondence between keys and colors.
2. *Graphic Mastermind :*
   Reinsert in the project the functions of random generation of a combination and of comparison of two combinations. Reprogram the main loop of the game using the graphic display.
3. *Last improvement :*
   We wish to be able to erase a color after its input in case of error. Study the functions `int keyboard();` and `void getCombination(int,int []);`
   The first one already takes care of the backspace key, but not the second one that considers it as a wrong key. Modify the latter in consequence.
4. In both projects, make the program display from the start the supposedly secret combination, so that your instructor can check your program more easily.

2. even if the function of combination analysis is not yet complete

3. Under Qt Creator, select it with the icon representing a screen on the left of the window.