SORBONNE UNIVERSITÉ

Autumn 2023

Password grid

TP 1

Aims

- ✯ Let familiarized with modern C++ syntax.
- ✯ Understanding the compiler

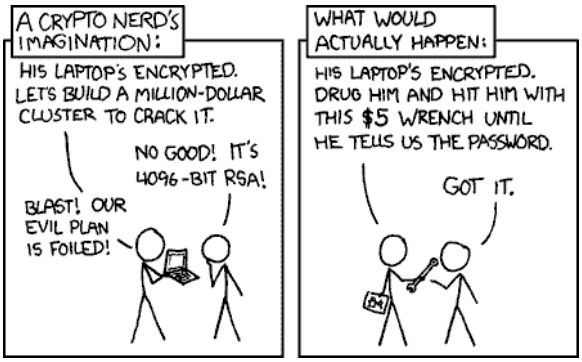
Constraints

- Indent your source files.
- Evaluation will take into account the briefness of the methods you wrote (avoid to write functions that are more than 25 line long) ; never hesitate to split a method into multiple shorter sub-methods (privates).
- Your code must not have errors in Valgrind (neither memory leak, nor other error).
- Vous ne devez pas utiliser de fonction C quand un équivalent C++ existe.
- For UML, you can use UMLet or Umbrello.
- Class names must start with a uppercase letter.
- You must provide a Makefile compile each source file and contains a rule called `clean` that delete intermediary files and a rule called `test` that execute the testcase binary.
- The **source and diagrams** must be *pushed* on your practical git.

Practical preparation

- A set of unit tests is provided in the folder `tests`, you can compile them with `make testcase`.
  The tests are using the library `doctest`, to learn more you can go on the project website `https://github.com/doctest/doctest`.
- The practical must be put on moodle at the end of the practical. It could be updated until the day before the next practical.

Concept

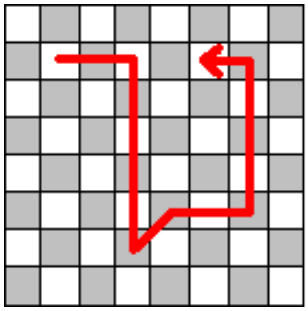


As xkcd's *strip* illustrates, obtaining a password is often easier than you might think. To increase security, we need to make it possible to forget the password and be unable to retrieve it. A simple technique is to use a password grid.

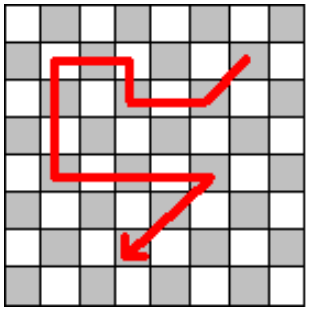
The principle of these grids is to generate a grid of random characters and define the password as a path on this grid. grid. In this way, the user memorizes not a sequence of characters but a path. Two ingredients are therefore required to access the protected system: the grid and the path. The grid is useless on its own as such (if the path is sufficiently complex); the path is also useless without the grid. If a user finds himself in the situation situation described in the previous strip, all he has to do is destroy the grid destroy the grid (e.g. burn it). He will then be physically unable to reveal his own password!

l	c	8	^	k	4	,	G
(	j	a	3	$	b	@	l
m	&	m	e	l	O	n	5
k	%	@	,	F	w	A	K
b	*	X	q	:	=	h	w
P	L	!	'	+	h	U	s
I	7	.	?	[	\	~	x
(	2	c	k	A	U	c	W

(a) An example of a password map.



(b) Path giving the password `6FVJ1vZ%ndTpCeq`.



(c) Path giving the password `eHYJVF6ixASvOcn%`.

1 Classe PassGrid

We want to implement a `PassGrid` class that represents a grid. grid. The constructor of this class takes as argument the size of the grid and randomly initializes each cell (we want displayable characters, i.e. with an ascii code between 33 and 94). In this first tutorial, you'll be asked to dynamic arrays and not STL containers. In addition to the copy, read accessors and destructor, the class must contain :

- a method `reset()` that randomly fills the grid.
- a method `save(std::ostream) const` that write the grid in a stream given as parameter.
- a method `print()` that write on `std::cout` the grid;
- a method `load(std::istream)` that loads a gridfrom a text file.
- An operator that read a cell (i,j) of the grid : `char operator()(size_t i, size_t j) const`

The random numbers can be generated with C function `rand()` (man 3 rand) or you can experience modern c++ library `random`.

The given unit tests check :

- The grid size
- The grid contains only printable characters
- That following grid generation are different
- That the method reset change the grid (This test supposes that a constructor by copy is implemented).

2 Path class

We now wish to create a `Path` class that represents a path in the grid. represents a path in the grid, as we need to generate random random passwords for users. There are 8 possible directions: N, NE. E, SE, S, SW, W, NW, which can be represented by integers. The class must contain :

- a constructor `Path(int n, size_t hmax, size_t wmaw)` which creates a random path of length `n` within a rectangle of height `hmax` and width `wmax`;
- a method `print()` which displays the path in a user-readable form (e.g. `(2,4) N-S-SW-NE`), where `(2,4)` is the starting point in the grid.
- a destructor accessors,
- an operator for accessing a cell (i) in the path

The unit tests provided verify :

- The initial cell is within the bounds
- The path remains in the grid.

3 Method generate()

Add to your class `PassGrid` a `generate` method that takes an object of type `Path` as an argument and returns a character chain representing the password.

4 Complete the tests (BONUS)

To familiarize yourself with unit testing, add the missing tests that show your program works.

5 Wrinting in a SVG file (BONUS)

Add an option to your program that takes a file name as an argument and writes the grid in SVG format to this file. You don't need to use C functions, but C++ file manipulation functions (see the `std::ofstream` class). For the SVG format, take a look at the documentation at `http://www.svgbasics.com/simple_text.html`. You can open SVGs with your web browser and edit them with inkscape.