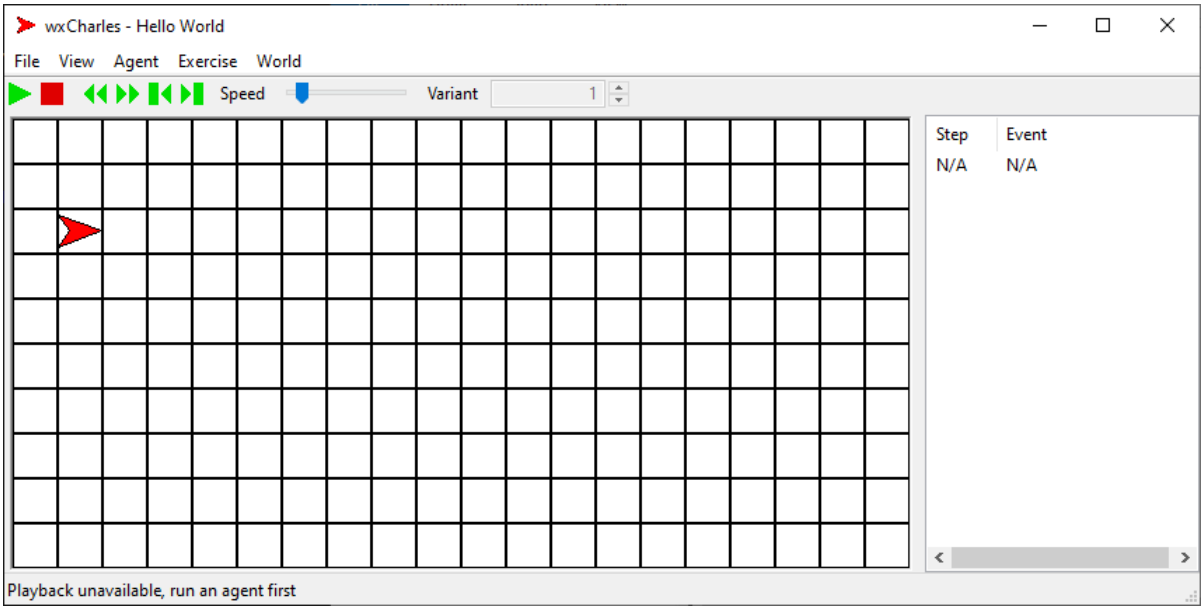
Assignment week 1: Charles the Robot

Course 'Imperative Programming' (IPC031)

Make sure to start working on the assignment **before the lab** session, otherwise you will be too late for the deadline

1 Background

Charles the Robot lives in a simple virtual world that consists of a plain rectangular area.



Charles' world can contain walls and balls. Charles can pick up a ball and put it in its pocket or take a ball out of its pocket and put it down at Charles' current location. The supply of balls is sufficiently large. Overall, Charles is a happy robot, but gets really upset when you want it to carry out impossible tasks:

- Charles can not exit his world.
- Charles can not walk through a wall.
- Charles can not pick up a ball if there is none at its current location.
- Charles can not place a ball if there is already a ball on that position.

When assigning tasks to Charles, you must take these rules into account.
Charles has a limited sense of its environment. It can detect whether it is right in front of a wall, stands above a ball, and if it is looking north.

2 Learning objectives

After doing this assignment you are able to:

- use the programming environment VSCode;
- use logical operations ('AND': `&&`, 'OR': `||`, and 'NOT': `!`) in choice and repetition;
- structure your algorithms by means of functions.

3 Instruction

On Brightspace you find instructions to install the recommended programming environment VSCode and the Charles program that you need for this week.

4 Assignment

In this assignment you design and implement a number of new tasks for Charles. Proceed as follows:

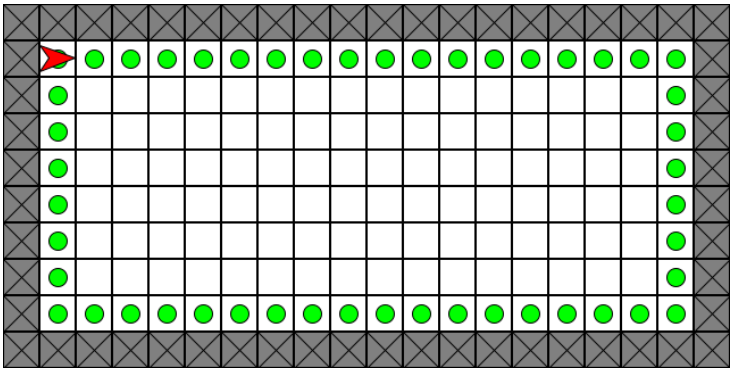
- For testing purposes, we have included the Charles command "Agent:Test". This command invokes the function `test_agent` (the body of this function is currently empty). In "`agent.cpp`" you can define the body of this function.
- First design your algorithms (problem solving phase). Structure your algorithms by means of functions.
- Take utmost care to give each function a name that indicates clearly its functionality ("what should it do?"). Use the naming convention that names of functions always start with a lower-case letter. If an adequate function name consists of several words, then they are connected with an `_` (underscore).
- Perform desktop tests for each and every function that you have designed. Ask yourself if it will perform correctly in every conceivable circumstance. If it cannot by design, then this should be documented (for instance, Charles can not step through a wall).
- Only when your design is complete and desktop tested, move on to the implementation phase. This can be done during the lab session of your choice.
- Add comments to your functions for documentation purposes. Even if a function has been given an expressive name, it may not be clear what the code in your function is doing.
- Compile your implementation to verify that it is syntactically correct. Test your implementation to verify that it is also semantically correct.

In this assignment you use only the following C++ language constructs and Charles constructs that have been explained in the lecture:

- C++: sequence (`;`), choice (`if-else`), conditional repetition (`while`), logical operations; functions;
- Charles: `step`, `turn_left`, `turn_right`, `get_ball`, `put_ball`, `in_front_of_wall`, `north`, `on_ball`.

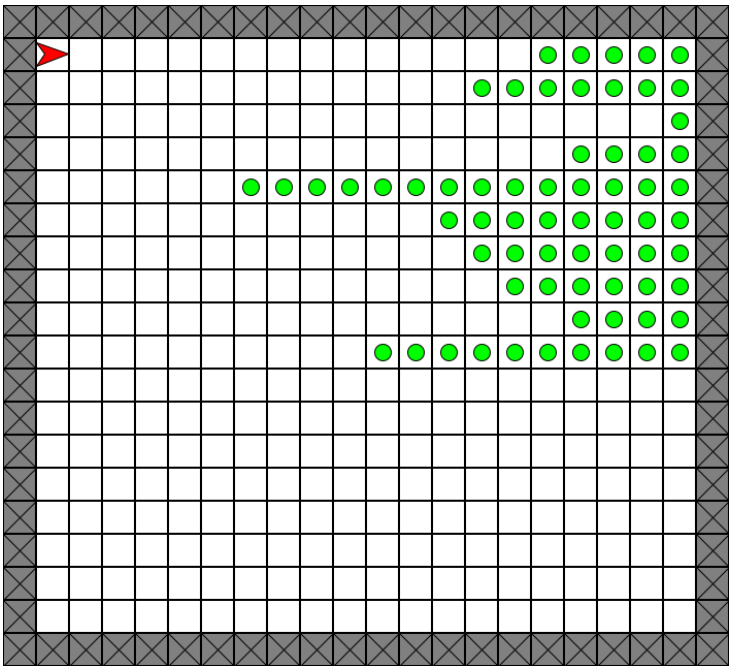
Below are the parts of the assignment that you are going to design and implement for Charles. You only need to modify "`agent.cpp`"! The content of all other files does not concern this exercise. Be aware that function-definition must precede function-use in the source code.

Part 1: Cleaning up a string of balls



Adjust the function `string_agent ()` in "`agent.cpp`". This function can be invoked by the Charles command "Exercise 1.1: cleaning up a string of balls". The effect of this function is that it makes Charles remove all balls that have been placed. All balls are placed 'along' the wall, and there are no holes. Charles is not allowed to 'leave' the wall. When done, Charles is back at its starting position, looking east.

Part 2: Cleaning up chaos with balls



Adjust the function `chaos_agent ()` in "`agent.cpp`". This function can be invoked by the Charles command "Exercise 1.2: cleaning up chaos with balls". The effect of this function is that it makes Charles remove all balls that have been placed at the 'east' wall. You can assume that each horizontal line of balls contains no holes. You can also assume that there are no 'empty' horizontal lines in between horizontal lines of balls. When done, Charles is back at its starting position, looking east. Your solution must work for any conceivable chaos of balls that meets the above criteria. Please note that a solution that makes Charles traverse his entire world is technically correct, but not considered decent (see lecture slide #24) because the solution is not proportional to the problem.

5 Products

As product-to-deliver you only need to upload to Brightspace the "`agent.cpp`" file that you have extended with solutions for each problem-solving-algorithm.

Deadline

Lab assignment: Friday, September 6, 2024, 23:59h

Important notes:

1. check that you have actually submitted your solution in Brightspace.
2. the deadline is firm, and it is impossible to upload a solution after expiry. In that case you fail the assignment.
3. you can upload solutions to Brightspace several times, but only the last submission is stored.
4. identify yourself and your lab partner in every uploaded document. The identification consists of your first and last names, student numbers, and number of (sub) assignment. By identifying yourself, you declare that the uploaded work has been created solely by you and your lab partner.
5. your work will be judged and commented upon. We expect that you obtain the feedback, read it, and use it to for the next exercises.
6. it is essential that you only submit your own solution, never copy somebody/something else's solution, and never share your solution—in particular: **AI tools (including but not limited to Github Copilot or ChatGPT) are not permitted**, solutions from previous year cannot be reused, and finally, you and your lab partner take joint responsibility for the assignment you submit.