Memory report

Project purpose

The purpose of this project is to create a no-contact drive-through system, which could be used during the Covid-19 times and beyond. The concept itself isn't advanced, but the implementation makes up for it. The program has a lot of input protections to ensure the proper working of the system, no matter the user-input. On top of this, a lot of time has been spent on making the system user-friendly, where it has adopted features which are like real-life self-service order interfaces. Finally, the variables and function names, use of header files and way of writing is very intuitive – together with a little knowledge of C++, it should be very simple to understand the working/purpose of the program in main.cpp.

Software requirements (specifications)

Software is written and compiled in Windows 10 in the ISO C++11 language standard (-std). The used IDE is Dev-C++ version 5.11. The compiler is TDM-GCC 4.9.2 64-bit release with mingw32 to make the Windows application. Standard C++ libraries were used, with functions that should be accessible on all supported platforms. However, this was **not** tested on MacOS or Linux systems.

Used programming elements

- Reading from and writing to files
- Pointers
- Vectors
- Arrays
- Dynamic memory
- Functions
- Header files
- Error handling/input protection
- Use of standard C++ libraries to promote cross-platform compatibility
- Use of time elements
    - Current date and time
    - Program pausing (which works on every OS, not just Windows)
- Interactive elements -> acts like a 'real' order system
    - Exit at any point in the program
    - Accidental exit protection
    - No loss of data (not having to restart the order if you clicked on exit and then returned to the order)
- Clear user interface with clear instructions and user feedback

I didn't know if it was obligatory to implement OOP for the project, and I contemplated to make an Order class where I could simulate having multiple orders of Order type, but then I would just create a class just to have one in my program (rather than actually needing it for the simulation). Therefore, I decided to not implement OOP in this program within the scope of this assignment. If it was necessary to use OOP then I apologise, but I hope you understand my reasoning for not using it in my project.

Process flowchart

The process flowchart of the drive-through order system can be seen in a simplified manner in Figure 1. The conditional based loops make for an interactive system, see for example the “Item?” block, where with the use of pointers the data in the food order never gets lost! This way of using pointers and (dynamic) vectors was new for me, and was a big learning point for me after lots of debugging and research.

Every input block has its own guard, which handles any other input than was intended. If this was not the case, the whole program would be able to crash in the first interaction. This wasn’t done with try-catch, for one because std::cin doesn’t throw an exception on bad input natively (although this can be enabled), but also to keep a better flow in the program, as exceptions should be used for exceptional situations – and I find the validation of user input not one of those exceptional situations.

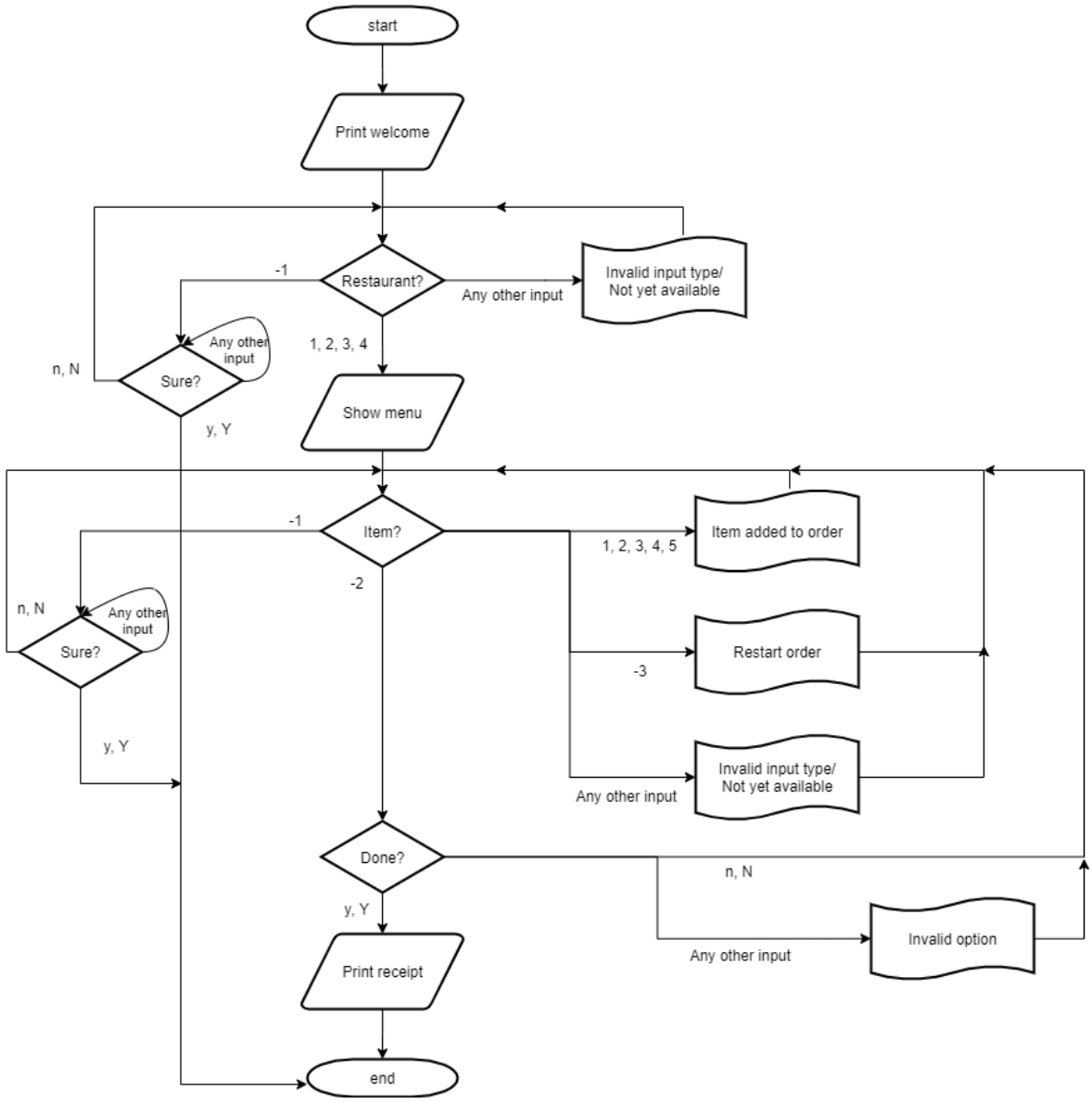

Figure 1: Process flowchart of the drive-through order system

Note: When running the executable, the “print receipt” text is not visible because the command prompt is being closed immediately after it is done. Check the order_receipts folder for the results!