Part 3 – Permit Software to be used with a range of models

To permit our software to be used with a range of washing machine models, we had to make sure our solution made it as easy as possible to allow additional washing programs to be added. With this in mind, we created a new header file, named **wash\_cycles.h**, which contained the description of the washing machine cycles as well as the class to store the wash program data and functionality.

We implemented 3 enumerated lists: eMotorControl{OFF, SLOW, FAST}, which defined the motor speeds available, eButtons{BUTONE, BUTTWO, BUTTHREE}, which defined the program buttons to be used \_\_\_\_\_\_\_\_\_\_ , and ePrograms{ProgWhite, ProgColour, TotalProgs}, which defined the available programs by ID.

In addition, we implemented struct wash\_cycle {char duration; char motor\_control; char status\_code}; which stored the data for each stage of the cycle, and

|  |
| --- |
| // Class to store the wash program data and functionality |
|  | class wash\_program { |
|  | public: |
|  | wash\_cycle get\_current\_wash\_cycle(); |
|  | wash\_program(wash\_cycle \*cycles, int length); |
|  | char wash\_length; |
|  | private: |
|  | wash\_cycle cycle\_array[MAX\_WASH\_LEN]; |
|  | }; |

which stored the wash program data and functionality.

Each wash cycle was expressed as an array of wash\_cycle structs, containing the duration, motor state and corresponding stage number for each stage of the wash cycle. For example, in the case of the colour cycles:

|  |
| --- |
| wash\_cycle colour\_cycles[] = { |
|  | 5,OFF,2, |
|  | 2,OFF,3, |
|  | 3,SLOW,4, |
|  | 4,OFF,1, |
|  | 4,OFF,2, |
|  | 4,SLOW,5, |
|  | 3,OFF,1, |
|  | 6,FAST,6, |
|  | 5,OFF,7, |
|  | 0,OFF,8 |
|  | }; |

Any new wash cycles could be added as a new array with the corresponding data.

Each wash\_cycle was used to instantiate a new wash\_program object, taking a pointer to the array containing the data and the number of stages in the program as arguments, then these wash\_program objects were added to an array, called program\_array[], which stored all available programs. Any new wash programs would also have to be instantiated then added to the array.

The void startWash(int prog\_num) took the desired program number and assigned the address of the wash\_program found at that position of the program\_array[] to a pointer pointing to the address of the current program, pCurrentProg.

The constructor function for wash\_program takes two arguments: 1. A pointer to the corresponding array wash\_cycle and 2. The number of cycles in the program (i.e. the length of the wash). It then fills the cycle\_array[] with the data from the wash\_cycle array passed in argument 1, stopping the for loop at the value corresponding to the length of the program.

The get\_current\_wash\_cycle() function returns the current stage of the wash cycle to be used in the main program.