Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

Computer Programming

Organizer

author: Maciej Krzyżowski
instructor: dr inż. Michał Marczyk
year: 2020/2021

Table of contents

1 Project's topic

A program should create lists (e.g. home, work, shopping) and enable the user to add tasks to the given list. The user should also remove the task from the list to mark the task as done.

2 User Interface

The program will display available lists with a number of tasks within given lists and prompt the user to input list's number to enter specific list. The user will also be given an option to exit the program. Upon choosing a list, its content will be shown with a short manual below containing commands that can be used. The tasks in the lists will be numbered and will have checkboxes next to them to display the task's status.

The user will be able to add, update or remove tasks by their index or remove all or finished tasks. Besides, there will also be an option to print the list again, print the list to file, rename the list, show the manual or change the current list.

3 Classes and Data Structures

3.1 Classes

The program contains four classes in total, two base and two derived classes, where Product inherits from class Task and ShoppingList inherits from class Organizer. Class Task represents a task which is stored in a list. It has a name, a description and a status. Class Product inherits the name and status from class Task but instead of description it has a cost.

Class Organizer contains a vector of objects of type Task, which acts as a list, and a name of the list. Class ShoppingList is an inherited class from the Organizer class with its vector storing objects of type Product.

Inheritance diagrams and methods' descriptions are in the Doxygen part of the report.

3.2 Data Structures

The data structure being used in the program is the vector, which stores objects being either tasks or products, depending of list's type. Using vectors allows having a data structure with variable size and easy adding, modifying and removing elements.

4 Algorithms

The program uses simple and not complicated algorithms. To make the program faster, it takes advantage of std::string_view instead of std::string in cases, where the string of text wouldn't be modified (e.g. when setting a new name for a task). This is because the std::string_view doesn't allocate memory as std::string does.

Exemplary method, which removes finished tasks:

```
void Organizer::removeFinished()
{
    int i = 0;
    for (auto x : this->list)
    {
        if (x.status)
        {
            this->list.erase(this->list.begin() + i);
        }
        i++;
    }
}
```

Exemplary method, which uses std::string_view:

```
void Product::set(std::string_view _name, double _cost)
{
    this->name = _name;
    this->cost = _cost;
    this->status = false;
}
```

The rest of the methods as well with overloaded functions and operators are also in the Doxygen part of the report. The program doesn't use global variables.

5 Program Overview

The program begins by prompting the user to choose a list from three available options. The user is then shown, which commands can be used in the program. The possible commands are:

- adding a new task or product
- updating its status
- removing it from the list
- printing the list either on the screen or to the file
- renaming the list
- printing list of available commands
- quitting the list

The program ends when while choosing a list the user inputs incorrect choice, which is told to the user at the beginning of the program.

The user is given the flexibility to use the program in many ways and in quick pace, with short and intuitive commands. The manual is easily accessible from within the program and gives detailed description of how the program can be used.

6 External Specification

The program has an option to output a chosen list to a text file. The text file has a name corresponding to the given list. The list in the file looks the same as in the console.

7 Testing

The program has been tested with various types of inputs. Incorrect ones are detected and an error message is printed. The program prevents the user from inputting a word where a number is required, which would cause the program to fail, and has protections needed with output file operations.

8 Conclusion

The program takes advantage of basic characteristics of object oriented programming, such as inheritance or polymorphism, as well as overloading. It also uses uncomplicated algorithms and focuses on fast execution of the code.

github link: `https://github.com/maciekrz/organizer`