ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ

за курсом «Основи Об’єктно-орієнтованого програмування»

студента групи ПС-20-1

Антонова Володимира Станіславовича

кафедра комп’ютерних технологій, ДНУ

2021/2022

*1. Постановка задачі*

**Загальні вимоги**

1. Реалізація методів ініціалізації (конструктор за замовчуванням та конструктор з параметрами), копіювання (конструктор копіювання), індексації (перевантаження []), присвоювання (перевантаження =), візуалізації, збереження (на диск) та відновлення, діалогового керування, "розумного доступу" (перевантаження ->), а також псевдозмінних (забезпечення можливості виду: f(x)=const).

2. Перевантаження (спільне використання) потокового введення/виведення (введення з файл, виведення в файл).

3. Створення та використання файла бібліотеки (\*.lib, або \*.obj).

4. Повторне використання класів без їх перекомпіляції (“ReUse”, тобто наслідування).

5. Застосування вказаної структури даних для розв’язання типової задачі.

В ході демонстрації роботи, програма дозволяє користувачеві в діалоговому режимі (за допомогою меню) виконувати наступні операції:

1. Створювати об’єкти класу (у довільній кількості).

2. Вводити/виводити об’єкти на диск.

3. Візуалізувати об’єкти.

4. Виконувати операції над об’єктами.

**Умова індивідуального завдання**

Стеки та обчислення виразів. Реалізувати розбір уведеного рядка для обчислення виразу. Вираз представляється в постфіксному вигляді, тобто ( 2 + 3) \*10

представляється у вигляді 23 + 10 \* . Розбір рядка виконується таким чином: числа заштовхуються в стек доки не зустрінеться символ операції, потім зі стеку виштовхуються два елемента, виконується операція, результат операції заштовхується в стек. Використання псевдозмінних для стеку може бути виконано таким чином: по-перше i-му елементу в стеці може бути присвоєно деяке значення, по-друге, наприклад, при

s(i,j)=s2; //де s, s2 – стеки

в стеку s починаючи з i-го символу j елементів замінюються на стек s2.

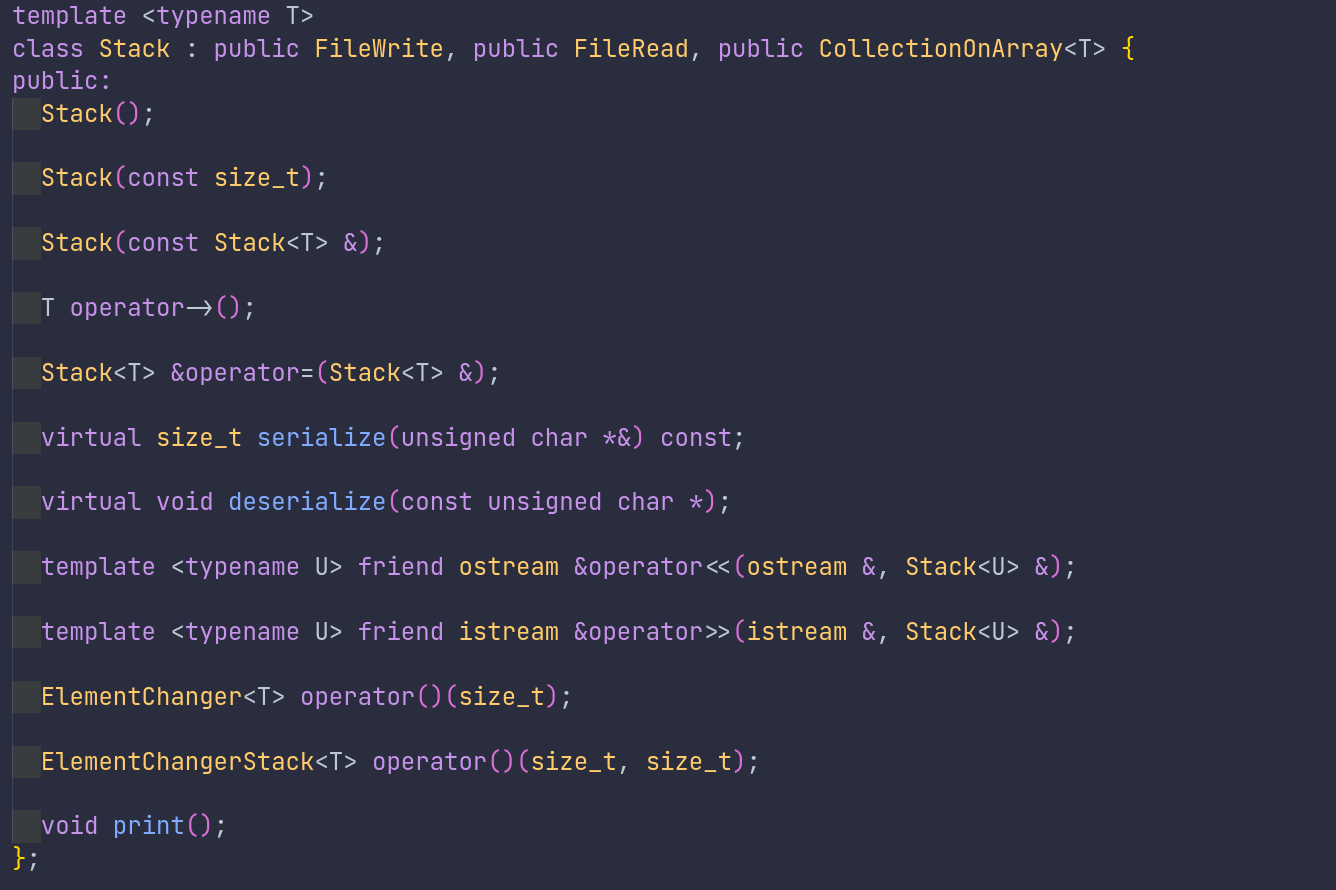
*2. Опис розв’язку*

У своєму завданні я маю працювати з виразами, для цього створюю клас Expression(валідує та виконує введенний вираз) та Stack(використовується як місце для введених виразів).

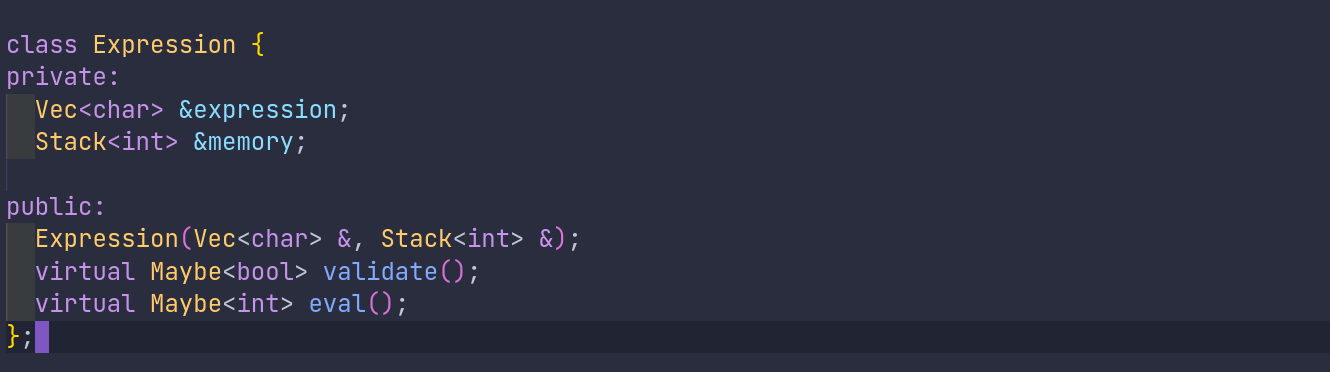
Для використання псевдозмінних створюю класи ElementChanger та ElementChangerStack.

У функції main, у циклі створюється меню за допомогою switch-case, щоб користувач міг обирати які дії хоче виконати.

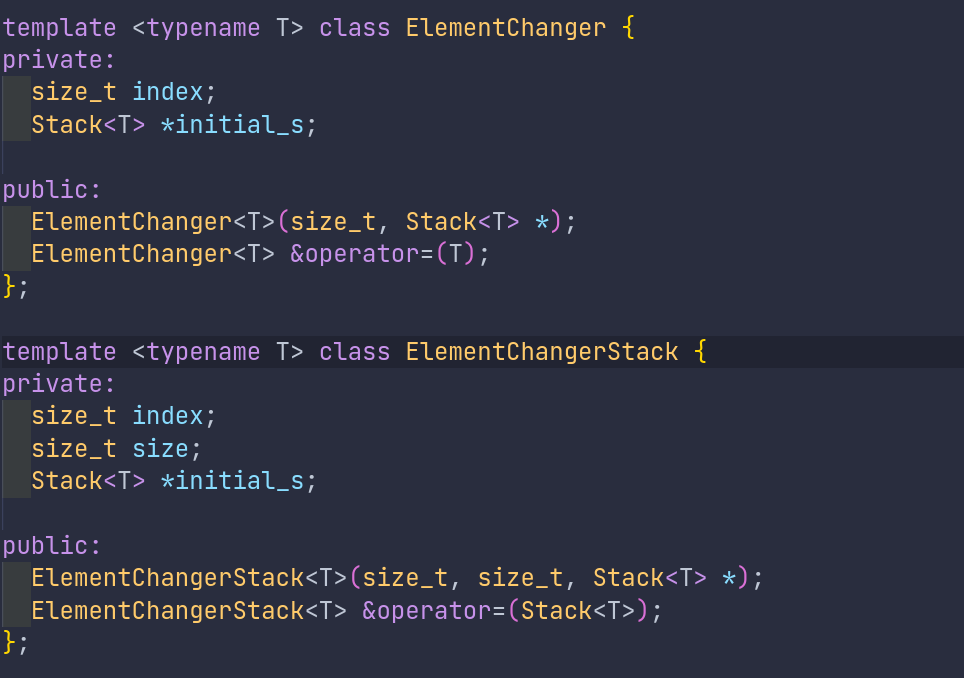
У класі Stack я описую конструктор за замовчуванням, конструктор з одним параметром для створення стеку з початковим розміром, конструктор копіювання, деструктор, також перевантаження [], =, ->, >>, <<, а також методи, для виконання дій з стеком.



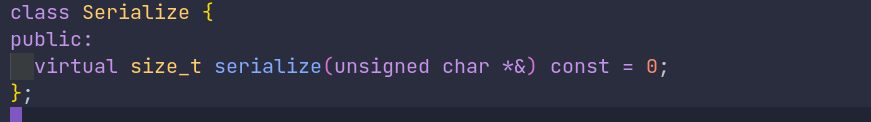
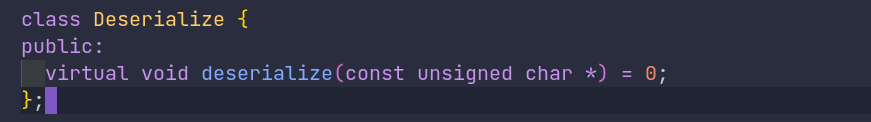
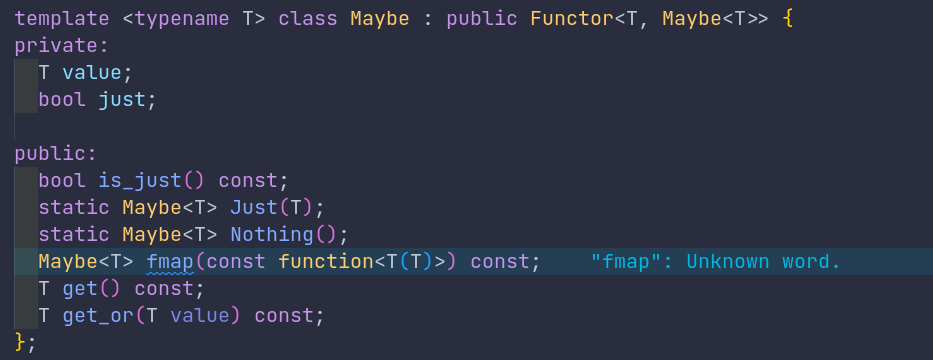
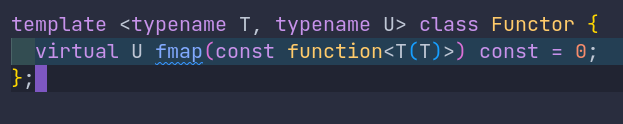
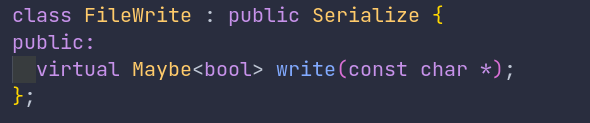
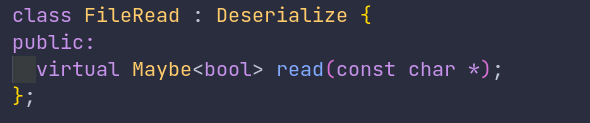
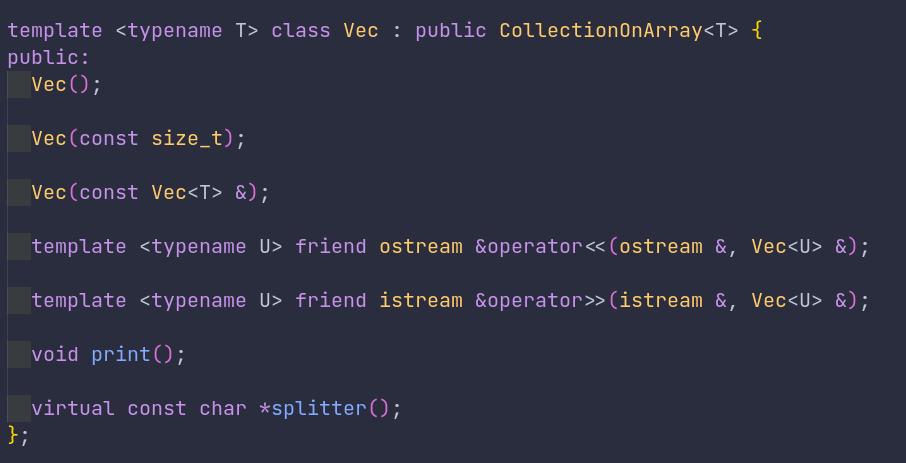
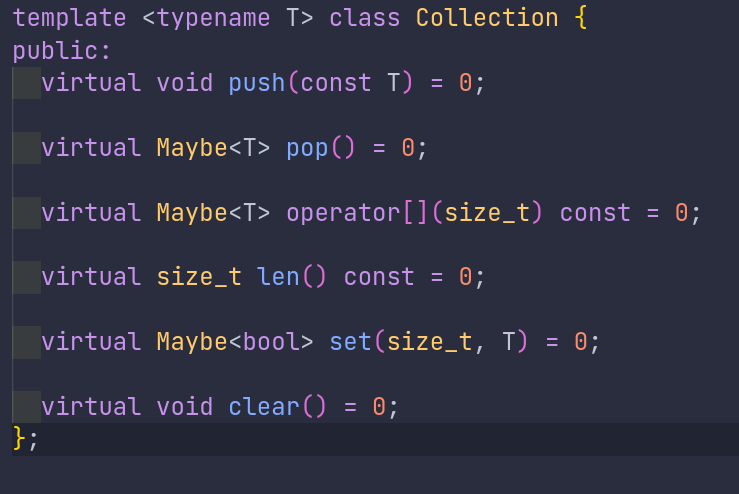
У класі Expression я описую конструктор з одним параметром для створення виразу



Для використання псевдозмінних створено клас ElementChanger та ElementChangerStack, у яких є конструктор та оператор =.



Також було створено допоміжні класи Collection, CollectionOnArray, Vec, FileRead, FileWrite, Functor, Maybe, Deserialize, Serialize.

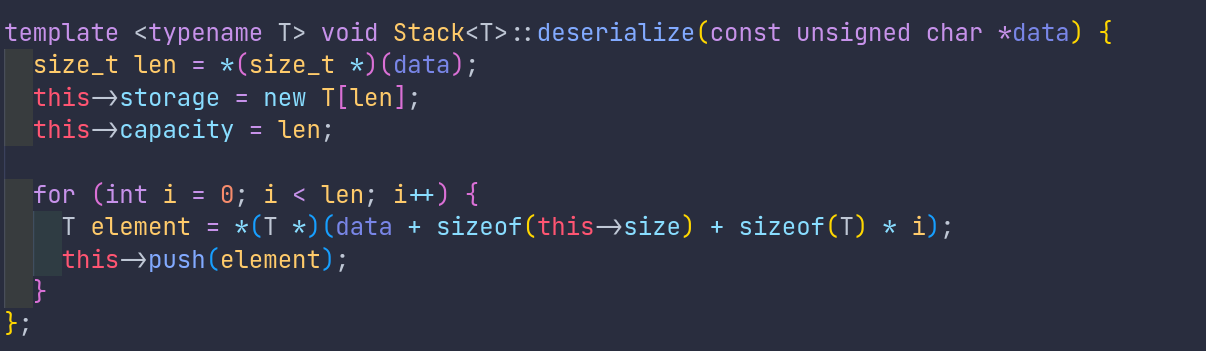


*3. Вихідний текст програми розв’язку задачі (основні фрагменти з коментарями)*

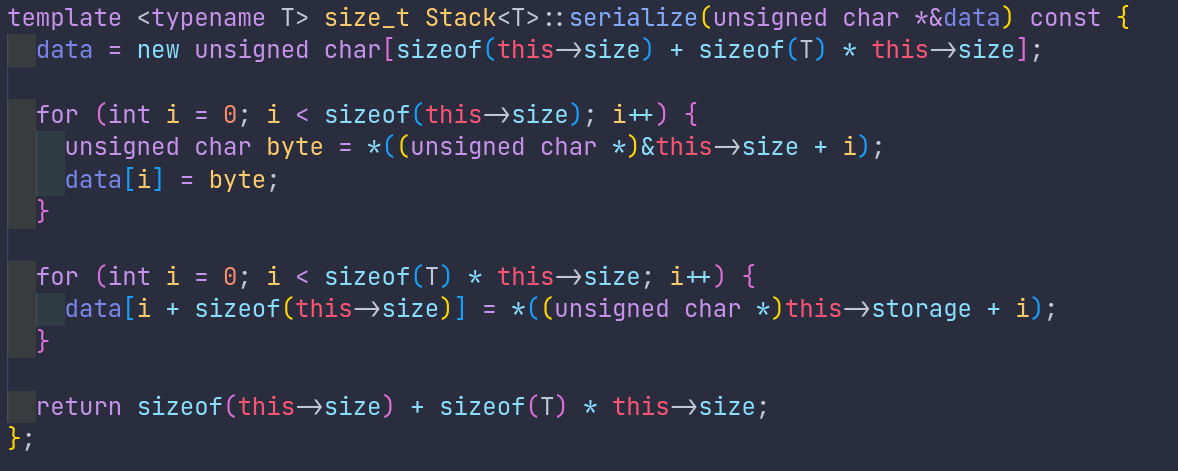
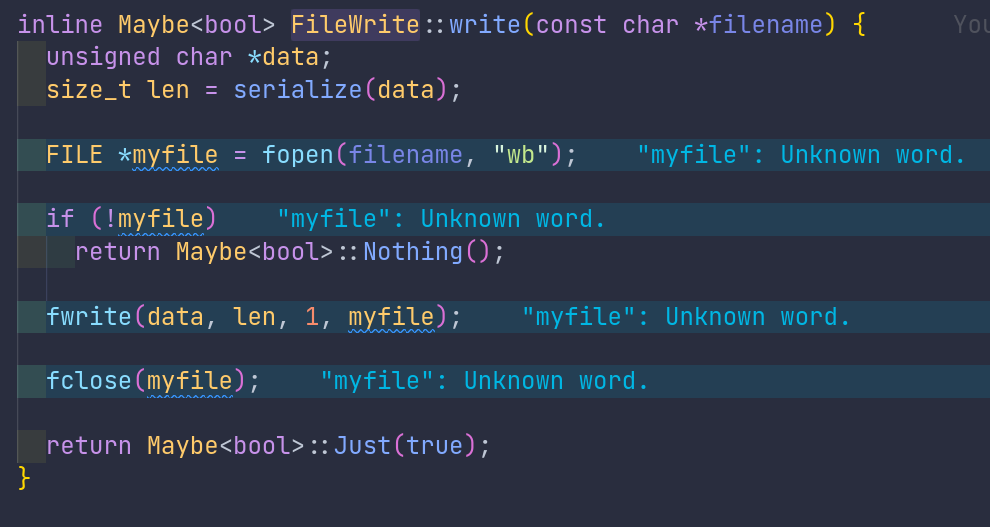
Клас Stack наслідується з класу CollectionOnArray у якому реалізовано методи() для колекцій, які базовано на масиві.



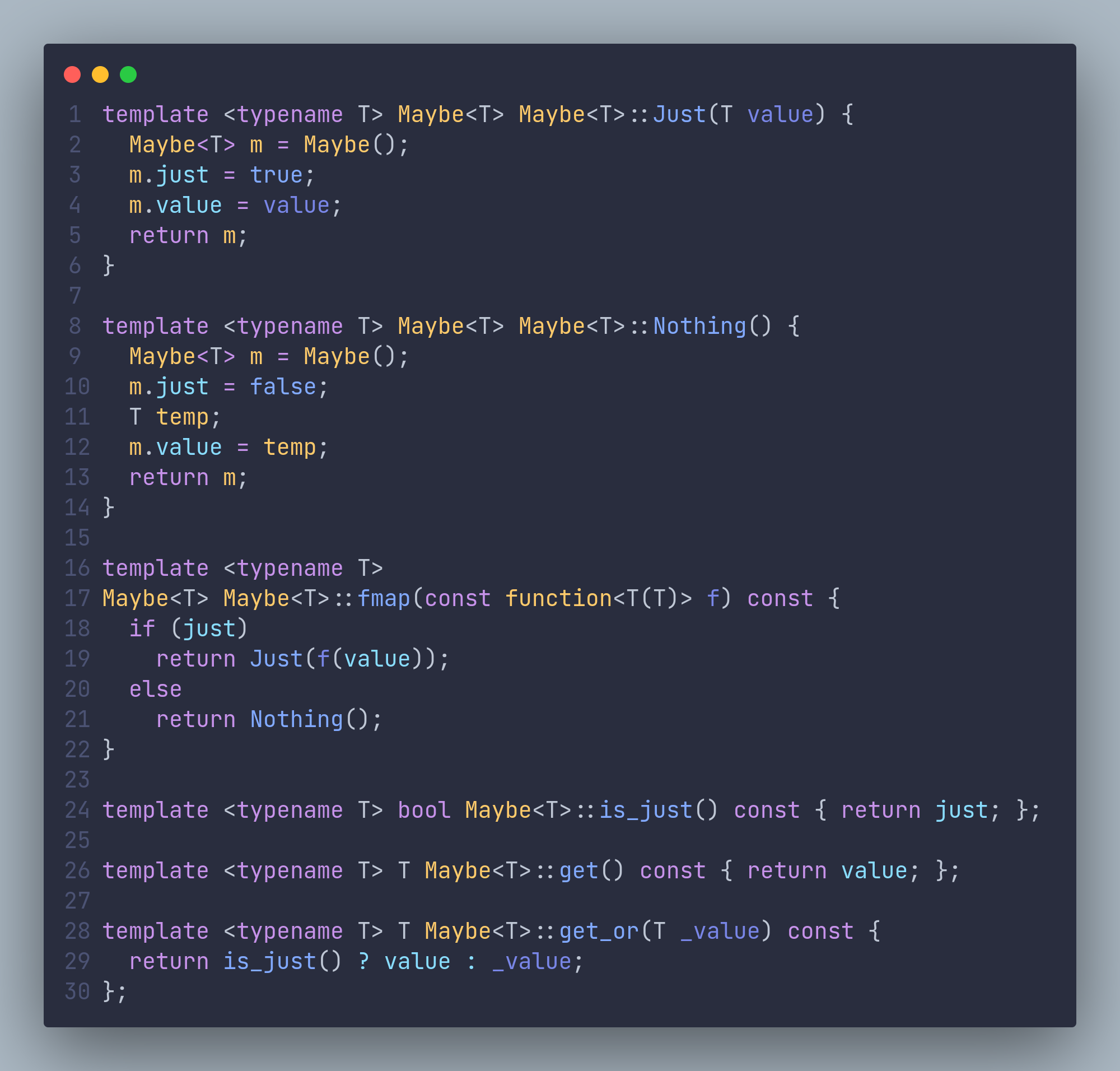
Для зчитування стеку з файлу використовуються класи FileRead та Deserialize:



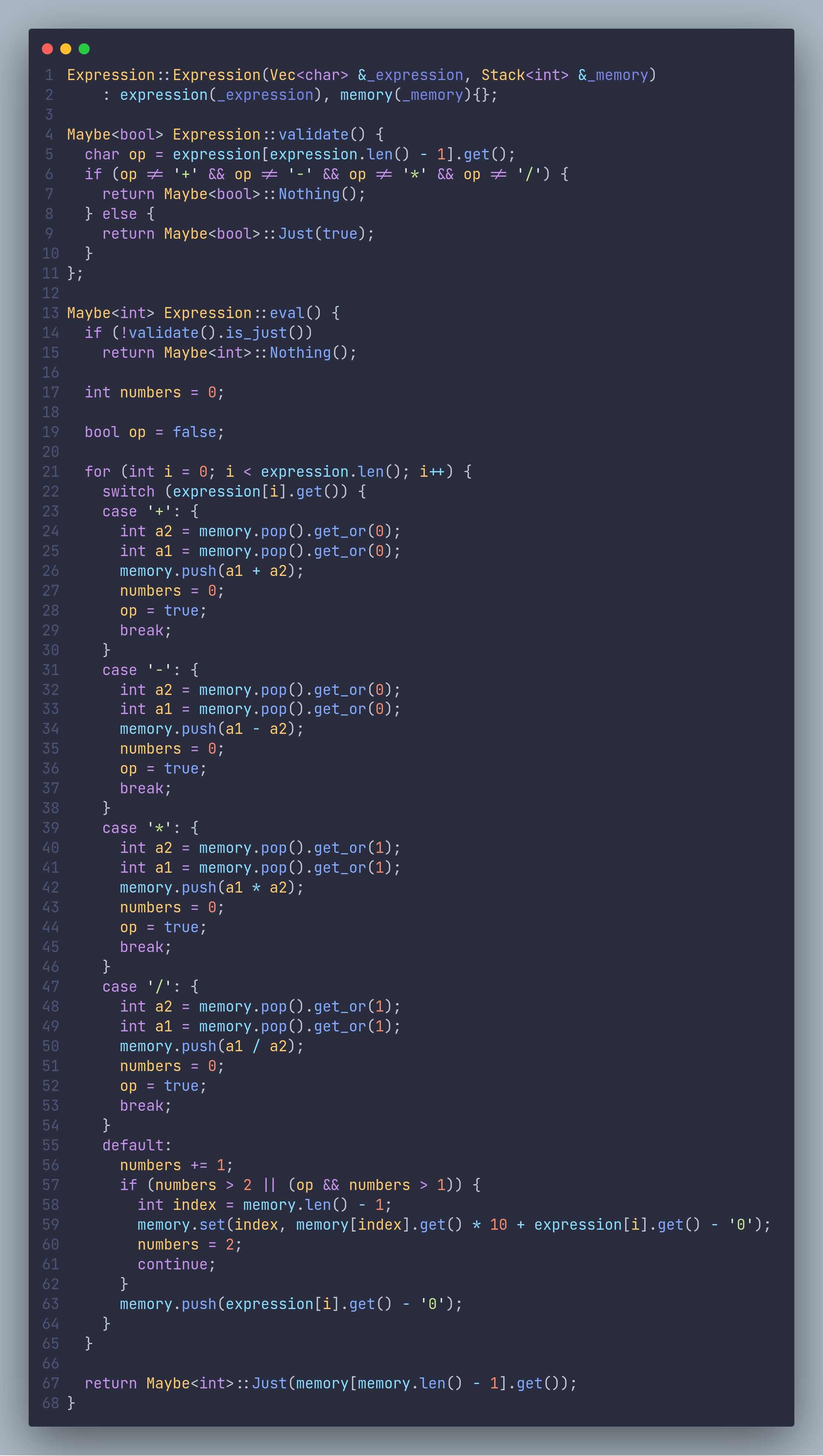
Для зберігання стеку у файлу використовуються класи FileWrite та Serialize:



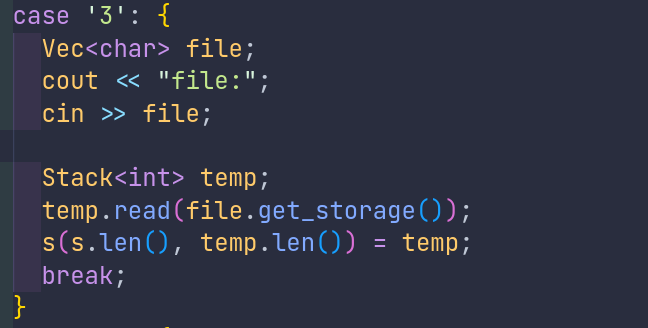
Для обробки помилок використовується клас Maybe:



Для обробки виразів використовується стек, спочатку якщо символів, то вони штовхаються у стек роздільно, якщо більше, то перший символ це перший елемент, а усі наступні – другий. Після першої дії, усі символи це один елемент.



Для зчитування файлу у стек використовуються псевдозмінні:

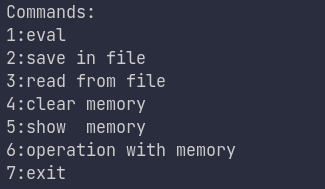


*4. Опис інтерфейсу програми (керівництво користувача)*

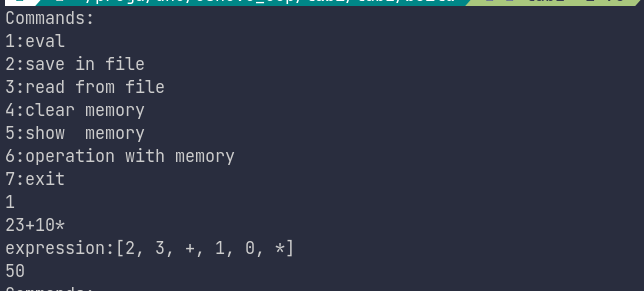
Для компілювання програми користувач може запустити build.sh. Програма буде розміщена у теці build.

Користувач може зробити 7 дій:

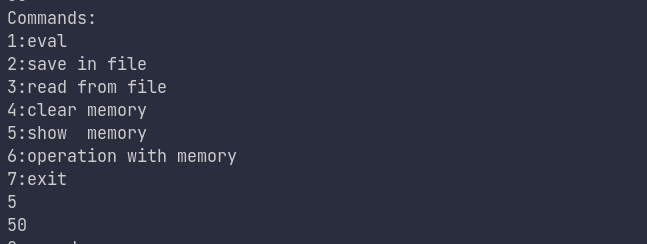
1. Порахувати вираз(ввод виразу)
2. Зберегти пам’ять у файл(ввод назви файлу)
3. Зчитати файл у пам’ять(ввод назви файлу)
4. Очистити пам’ять
5. Вивести пам’ять
6. Зробити операцію з двома останніми елементами у пам’яті(ввод операції)
7. Завершити програму



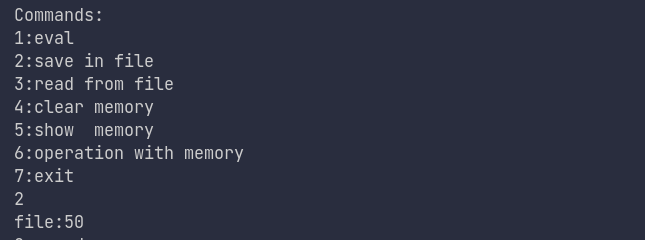
Виконання виразу



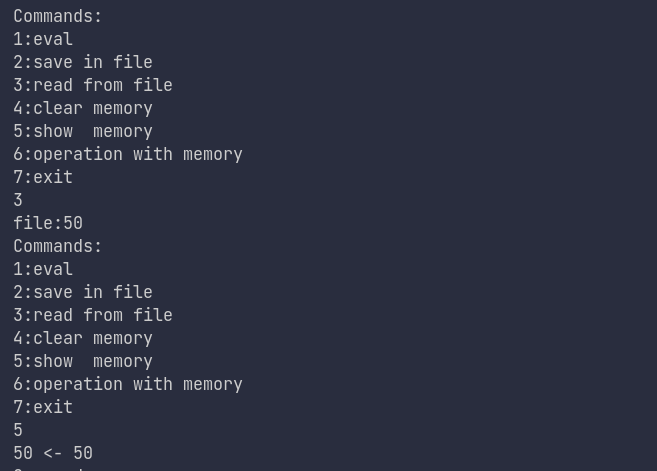
Перегляд пам’яті



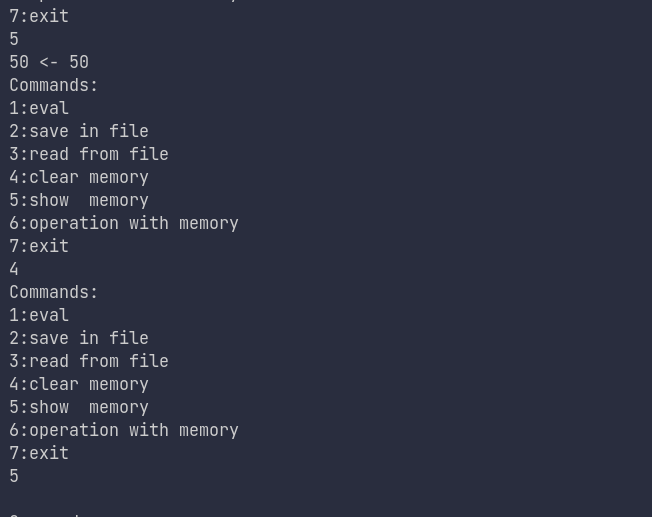
Збереження у файл



Зчитування з файлу



Видалення в пам’яті



Операція з елементами в пам’яті

