Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»

Лабораторная работа №8

Выполнил:
студенты 4 курса
группы ИП-216
Андрущенко Ф.А.
Литвинов А. Е.
Русецкий А. С.

Проверил:
преподаватель кафедры ПМиК
Агалаков Антон Александрович

Новосибирск, 2025 г.

Задание

1. В соответствии с приведенной ниже спецификацией реализовать параметризованный абстрактный тип данных «память», для хранения одного числа – объекта типа T, используя шаблон классов C++
2. Протестировать каждую операцию, определенную на типе данных, используя средства модульного тестирования
3. Если необходимо, предусмотрите возбуждение исключительных ситуаций

Спецификация типа данных Процессор».
ADT TProc
Данные
Процессор (тип TProc) выполняет двухоперандные операции TOprtn = (None, Add, Sub, Mul, Dvd) и однооперандные операции - функции TFunc = (Rev, Sqr) над значениями типа T. Левый операнд и результат операции хранится в поле Lop_Res, правый - в поле Rop. Оба поля имеют тип T. Процессор может находиться в состояниях: «операция установлена» - поле Operation не равно None (значение типа TOprtn) или в состоянии «операция не установлена» - поле Operation = None. Значения типа TProc - изменяемые. Они изменяются операциями: «Сброс операции» (OprtnClear), «Выполнить операцию» (OprtnRun), «Вычислить функцию» (FuncRun), «Установить операцию» (OprtnSet), «Установить левый операнд» (Lop_Res_Set), «Установить правый операнд» (Rop_Set), «Сброс калькулятора» (ReSet). На значениях типа T должны быть определены указанные выше операции и функции.

Операции

Конструктор	
Начальные значения:	Нет
Процесс:	Инициализирует поля объекта процессор типа TProc. Поля Lop_Res, Rop инициализируются объектами (тип T) со значениями по умолчанию. Например, для простых дробей - 0/1. Процессор устанавливается в состояние: «операция не установлена»: (Operation = None).

СбросПроцессора	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Поля объекта процессор: Lop_Res, Rop инициализируются объектами (тип T) со значениями по умолчанию. Например, для простых дробей - 0/1. Процессор устанавливается в состояние: «операция не установлена»: (Operation = None).
Выход:	Нет.
Постусловия:	Состояние процессора – «операция сброшена» (Operation = None).
СбросОперации	
Вход:	Нет.
Предусловия:	Нет.
Процесс	Процессор устанавливается в состояние: «операция не установлена»: (Operation = None).
Выход:	Нет.
Постусловия:	Состояние процессора – «операция сброшена» (Operation = None).
ВыполнитьОперацию	
Вход:	Нет.
Предусловия:	Нет.
Процесс	Вызывает выполнение текущей операции (записанной в поле Operation). Операция (Operation) выполняется над значениями, хранящимися в полях Rop и Lop_Res. Результат сохраняется в поле Lop_Res. Если Operation = None, никакие действия не выполняются. Состояние объекта не изменяется.
Выход:	Нет.
Постусловия:	Состояние процессора не изменяется.
ВычислитьФункцию	
Вход:	Вид функции (Func: TFunc).
Предусловия:	Нет.
Процесс	Вызывает выполнение текущей функции (Func). Функция (Func) выполняется над значением, хранящимся в поле Rop. Результат сохраняется в нём же. Состояние объекта не изменяется.
Выход:	Нет.

Постусловия:	Состояние процессора не меняется.
ЧитатьЛевыйОперанд	
Вход:	Нет.
Предусловия:	Нет.
Процесс	Создаёт и возвращает копию объекта, который хранится в поле Lop_Res.
Выход:	Объект типа T.
Постусловия:	Состояние процессора не изменяется.
ЗаписатьЛевыйОперанд	
Вход:	Переменная Operand типа T.
Предусловия:	Нет.
Процесс	Создаёт копию объекта Operand и заносит её в поле Lop_Res.
Выход:	Нет.
Постусловия:	Состояние процессора не изменяется.
ЧитатьПравыйОперанд	
Вход:	Нет.
Предусловия:	Нет.
Процесс	Создаёт и возвращает копию объекта, который хранится в Rop.
Выход:	Объект типа T.
Постусловия:	Состояние процессора не меняется.
ЗаписатьПравыйОперанд	
Вход:	Переменная Operand типа T.
Предусловия:	Нет.
Процесс	Создаёт копию объекта Operand и заносит её в поле Rop.
Выход:	Нет.
Постусловия:	Состояние процессора не изменяется.
ЧитатьСостояние	
Вход:	Нет.
Предусловия:	Нет.
Процесс	Копирует и возвращает значение поля Operation.
Выход:	Значение поля Operation.
Постусловия:	Состояние процессора не изменяется.
ЗаписатьСостояние	
Вход:	Переменная Oprtn типа TOprtn.
Предусловия:	Нет.
Процесс	Заносит значение Oprtn в поле Operation.
Выход:	Нет.
Постусловия:	Состояние процессора изменяется на Oprtn.

Исходный код программы

```
from typing import TypeVar, Generic, Optional, Union
from enum import Enum, auto

T = TypeVar('T')


class TOprtn(Enum):
    """Тип операций"""
    None_ = auto()
    Add = auto()
    Sub = auto()
    Mul = auto()
    Dvd = auto()


class TFunc(Enum):
    """Тип функций"""
    Rev = auto()
    Sqr = auto()


class TProc(Generic[T]):
    """Параметризованный класс Процессор для выполнения операций над значениями типа
T"""

    def __init__(self, default_value: T):
        """
        Конструктор
        Начальные значения:
        - Lop_Res: объект типа T со значением по умолчанию
        - Rop: объект типа T со значением по умолчанию
        - Operation: None_ (операция не установлена)
        """
        self._Lop_Res = default_value
        self._Rop = default_value
        self._Operation = TOprtn.None_

    def ResetProcessor(self, default_value: T) -> None:
        """
        СбросПроцессора
        Процесс: Сбрасывает все поля к значениям по умолчанию
        """
        self._Lop_Res = default_value
        self._Rop = default_value
        self._Operation = TOprtn.None_

    def ResetOperation(self) -> None:
        """
        СбросОперации
```

```
        Процесс: Сбрасывает только операцию
        """
        self._Operation = TOprtn.None_

    def ExecuteOperation(self) -> None:
        """
        ВыполнитьОперацию
        Процесс: Выполняет текущую операцию над Lop_Res и Rop, результат сохраняет в
Lop_Res
        """
        if self._Operation == TOprtn.None_:
            return

        try:
            if self._Operation == TOprtn.Add:
                self._Lop_Res = self._Lop_Res + self._Rop
            elif self._Operation == TOprtn.Sub:
                self._Lop_Res = self._Lop_Res - self._Rop
            elif self._Operation == TOprtn.Mul:
                self._Lop_Res = self._Lop_Res * self._Rop
            elif self._Operation == TOprtn.Dvd:
                self._Lop_Res = self._Lop_Res / self._Rop
        except ZeroDivisionError:
            raise ZeroDivisionError("Деление на ноль")
        except TypeError as e:
            raise TypeError(
                f"Операция не поддерживается для типа {type(self._Lop_Res).__name__}"
            ) from e

    def ExecuteFunction(self, func: TFunc) -> None:
        """
        ВычислитьФункцию
        Вход: func - тип функции
        Процесс: Выполняет функцию над Rop, результат сохраняет в Rop
        """
        try:
            if func == TFunc.Rev:
                self._Rop = 1 / self._Rop
            elif func == TFunc.Sqr:
                self._Rop = self._Rop * self._Rop
        except ZeroDivisionError:
            raise ZeroDivisionError("Обращение нуля")
        except TypeError as e:
            raise TypeError(
                f"Функция не поддерживается для типа {type(self._Rop).__name__}"
            ) from e

    def ReadLeftOperand(self) -> T:
        """
        ЧитатьЛевыйОперанд
        Выход: копия левого операнда
        """
```

```
        return self._Lop_Res

    def WriteLeftOperand(self, operand: T) -> None:
        """
        ЗаписатьЛевыйОперанд
        Вход: operand - объект типа T
        Процесс: Записывает копию operand в Lop_Res
        """
        self._Lop_Res = operand

    def ReadRightOperand(self) -> T:
        """
        ЧитатьПравыйОперанд
        Выход: копия правого операнда
        """
        return self._Rop

    def WriteRightOperand(self, operand: T) -> None:
        """
        ЗаписатьПравыйОперанд
        Вход: operand - объект типа T
        Процесс: Записывает копию operand в Rop
        """
        self._Rop = operand

    def ReadState(self) -> TOprtn:
        """
        ЧитатьСостояние
        Выход: текущая операция
        """
        return self._Operation

    def WriteState(self, oprtn: TOprtn) -> None:
        """
        ЗаписатьСостояние
        Вход: oprtn - тип операции
        Процесс: Устанавливает новую операцию
        """
        self._Operation = oprtn

    @property
    def Lop_Res(self) -> T:
        """Свойство для чтения/записи левого операнда-результата"""
        return self._Lop_Res

    @Lop_Res.setter
    def Lop_Res(self, value: T) -> None:
        self._Lop_Res = value

    @property
    def Rop(self) -> T:
        """Свойство для чтения/записи правого операнда"""
```

```
        return self._Rop

    @Rop.setter
    def Rop(self, value: T) -> None:
        self._Rop = value

    @property
    def Operation(self) -> TOprtn:
        """Свойство для чтения/записи операции"""
        return self._Operation

    @Operation.setter
    def Operation(self, value: TOprtn) -> None:
        self._Operation = value

    def __str__(self) -> str:
        """Строковое представление процессора"""
        op_map = {
            TOprtn.None_: "None",
            TOprtn.Add: "+",
            TOprtn.Sub: "-",
            TOprtn.Mul: "*",
            TOprtn.Dvd: "/"
        }
        return f"Процессор[Lop_Res: {self._Lop_Res}, Rop: {self._Rop}, Operation:
{op_map[self._Operation]}]"


if __name__ == "__main__":
    proc = TProc[int](0)
    print(f"Создан процессор: {proc}")

    proc.WriteLeftOperand(10)
    proc.WriteRightOperand(5)
    print(f"После установки операндов: {proc}")

    proc.WriteState(TOprtn.Add)
    print(f"После установки операции: {proc}")

    proc.ExecuteOperation()
    print(f"После выполнения операции: {proc}")

    proc.WriteRightOperand(4)
    proc.ExecuteFunction(TFunc.Sqr)
    print(f"После вычисления квадрата: {proc}")

    proc.ResetProcessor(0)
    print(f"После сброса: {proc}")
```

Модульные тесты для тестирования класса

```python
import unittest
from typing import List
from tprocessor import TProc, TOprtn, TFunc


class TestTProc(unittest.TestCase):
    """Тесты для класса TProc"""

    def test_initialization(self):
        """Тест инициализации"""
        proc = TProc[int](0)
        self.assertEqual(proc.Lop_Res, 0)
        self.assertEqual(proc.Rop, 0)
        self.assertEqual(proc.Operation, TOprtn.None_)
        self.assertEqual(proc.ReadState(), TOprtn.None_)

    def test_reset_processor(self):
        """Тест сброса процессора"""
        proc = TProc[int](0)
        proc.WriteLeftOperand(10)
        proc.WriteRightOperand(5)
        proc.WriteState(TOprtn.Add)
        proc.ResetProcessor(0)
        self.assertEqual(proc.Lop_Res, 0)
        self.assertEqual(proc.Rop, 0)
        self.assertEqual(proc.Operation, TOprtn.None_)

    def test_reset_operation(self):
        """Тест сброса операции"""
        proc = TProc[int](0)
        proc.WriteState(TOprtn.Add)
        proc.ResetOperation()
        self.assertEqual(proc.Operation, TOprtn.None_)

    def test_execute_operation_add(self):
        """Тест выполнения операции сложения"""
        proc = TProc[int](0)
        proc.WriteLeftOperand(10)
        proc.WriteRightOperand(5)
        proc.WriteState(TOprtn.Add)
        proc.ExecuteOperation()
        self.assertEqual(proc.Lop_Res, 15)
        self.assertEqual(proc.Rop, 5)

    def test_execute_operation_sub(self):
        """Тест выполнения операции вычитания"""
        proc = TProc[int](0)
        proc.WriteLeftOperand(10)
        proc.WriteRightOperand(5)
```

```
        proc.WriteState(TOprtn.Sub)
        proc.ExecuteOperation()
        self.assertEqual(proc.Lop_Res, 5)

    def test_execute_operation_mul(self):
        """Тест выполнения операции умножения"""
        proc = TProc[int](0)
        proc.WriteLeftOperand(10)
        proc.WriteRightOperand(5)
        proc.WriteState(TOprtn.Mul)
        proc.ExecuteOperation()
        self.assertEqual(proc.Lop_Res, 50)

    def test_execute_operation_div(self):
        """Тест выполнения операции деления"""
        proc = TProc[float](0.0)
        proc.WriteLeftOperand(10.0)
        proc.WriteRightOperand(5.0)
        proc.WriteState(TOprtn.Dvd)
        proc.ExecuteOperation()
        self.assertEqual(proc.Lop_Res, 2.0)

    def test_execute_operation_none(self):
        """Тест выполнения операции None (ничего не должно происходить)"""
        proc = TProc[int](0)
        proc.WriteLeftOperand(10)
        proc.WriteRightOperand(5)
        proc.ExecuteOperation()
        self.assertEqual(proc.Lop_Res, 10)

    def test_execute_function_sqr(self):
        """Тест выполнения функции квадрата"""
        proc = TProc[int](0)
        proc.WriteRightOperand(5)
        proc.ExecuteFunction(TFunc.Sqr)
        self.assertEqual(proc.Rop, 25)

    def test_operand_read_write(self):
        """Тест чтения/записи операндов"""
        proc = TProc[int](0)
        proc.WriteLeftOperand(100)
        proc.WriteRightOperand(200)
        self.assertEqual(proc.ReadLeftOperand(), 100)
        self.assertEqual(proc.ReadRightOperand(), 200)

    def test_state_read_write(self):
        """Тест чтения/записи состояния"""
        proc = TProc[int](0)
        proc.WriteState(TOprtn.Mul)
        self.assertEqual(proc.ReadState(), TOprtn.Mul)

    def test_properties(self):
```

```
        """Тест свойств"""
        proc = TProc[int](0)
        proc.Lop_Res = 123
        proc.Rop = 456
        proc.Operation = TOprtn.Sub
        self.assertEqual(proc.Lop_Res, 123)
        self.assertEqual(proc.Rop, 456)
        self.assertEqual(proc.Operation, TOprtn.Sub)


class TestTProcWithList(unittest.TestCase):
    """Тесты для TProc с пользовательским типом List"""

    def test_list_operations(self):
        """Тест операций со списками"""
        proc = TProc[List[int]]([])
        proc.WriteLeftOperand([1, 2])
        proc.WriteRightOperand([3, 4])
        proc.WriteState(TOprtn.Add)
        proc.ExecuteOperation()
        self.assertEqual(proc.Lop_Res, [1, 2, 3, 4])

        with self.assertRaises(TypeError):
            proc.ExecuteFunction(TFunc.Sqr)


if __name__ == "__main__":
    from rich import print
    from rich.panel import Panel
    from rich.console import Console
    from unittest import TextTestRunner, TestResult, TestSuite

    console = Console()

    class RichTestResult(TestResult):
        def __init__(self, stream, descriptions, verbosity):
            super().__init__(stream, descriptions, verbosity)

        def startTest(self, test):
            super().startTest(test)
            console.print(f"[cyan]Running:[/cyan] {test._testMethodName}")

        def addSuccess(self, test):
            super().addSuccess(test)
            console.print(f"[green]✓ PASS:[/green] {test._testMethodName}")

        def addFailure(self, test, err):
            super().addFailure(test, err)
            console.print(f"[red]✗ FAIL:[/red] {test._testMethodName}")

        def addError(self, test, err):
            super().addError(test, err)
```

```
            console.print(f"[magenta]💥 ERROR:[/magenta] {test._testMethodName}")

    loader = unittest.defaultTestLoader
    suite = TestSuite()
    suite.addTests(loader.loadTestsFromTestCase(TestTProc))
    suite.addTests(loader.loadTestsFromTestCase(TestTProcWithList))

    runner = TextTestRunner(resultclass=RichTestResult, verbosity=0)
    result = runner.run(suite)

    console.print(
        Panel.fit(
            f"[green]Passed: {result.testsRun - len(result.failures) -
len(result.errors)}[/green]\n"
            f"[red]Failed: {len(result.failures)}[/red]\n"
            f"[magenta]Errors: {len(result.errors)}[/magenta]\n"
            f"[yellow]Total: {result.testsRun}[/yellow]",
            title="Test Results",
        )
    )
```

Результат тестирования методов класса

```
❯ python test_tprocessor.py
Running: test_execute_function_sqr
✓ PASS: test_execute_function_sqr
Running: test_execute_operation_add
✓ PASS: test_execute_operation_add
Running: test_execute_operation_div
✓ PASS: test_execute_operation_div
Running: test_execute_operation_mul
✓ PASS: test_execute_operation_mul
Running: test_execute_operation_none
✓ PASS: test_execute_operation_none
Running: test_execute_operation_sub
✓ PASS: test_execute_operation_sub
Running: test_initialization
✓ PASS: test_initialization
Running: test_operand_read_write
✓ PASS: test_operand_read_write
Running: test_properties
✓ PASS: test_properties
Running: test_reset_operation
✓ PASS: test_reset_operation
Running: test_reset_processor
✓ PASS: test_reset_processor
Running: test_state_read_write
✓ PASS: test_state_read_write
Running: test_list_operations
✓ PASS: test_list_operations
Ran 13 tests in 0.005s

OK
┌─ Test Results ─┐
  Passed: 13
  Failed: 0
  Errors: 0
  Total: 13
```