Міністерство освіти і науки України
Львівський національний університет імені Івана Франка Факультет електроніки та комп’ютерних технологій

Звіт

про виконання лабораторної роботи №9

з курсу “Функціональне програмування ”

“**Помилки як дані: Maybe / Result (Either-патерн)**”

Виконав:
студент групи ФЕП-23
Чепара Станіслав
Перевірив:
доцент Франів В. А.

Львів 2025

Мета роботи: відмова від винятків у доменній логіці; робота з комбінаторами map / flat_map (and_then) / map_err; побудова конвеєра перетворень із коротким замиканням на першій помилці.

Хід роботи

1. **ADT для помилок: Maybe = Some[T] | Nothing, Result = Ok[T] | Err[E] з імутабельними @dataclass.**
   lab09_result_maybe.py

```python
# -------- Maybe (Option) --------
@dataclass(frozen=True, slots=True)
class Some(Generic[T]):
    value: T

@dataclass(frozen=True, slots=True)
class Nothing:
    pass

Maybe: TypeAlias = Some[T] | Nothing  # параметризований тип-аліас


# -------- Result (Either) --------
@dataclass(frozen=True, slots=True)
class Ok(Generic[T]):
    value: T

@dataclass(frozen=True, slots=True)
class Err(Generic[E]):
    error: E

Result: TypeAlias = Ok[T] | Err[E]  # параметризований тип-аліас
```

2. Комбінатори: map_result, and_then (flat_map), map_err, unwrap_or + версії для Maybe.

lab09_result_maybe.py

```python
# B) Комбінатори Result/Maybe
# ===========================

def map_result(r: Result[T, E], fn: Callable[[T], U]) -> Result[U, E]:
    """Якщо Ok — застосувати fn до value, інакше повернути Err як є."""
    if isinstance(r, Ok):
        return Ok(fn(r.value))
    return r  # type: ignore[return-value]  # Err[E] сумісний з Result[U, E]


def and_then(r: Result[T, E], fn: Callable[[T], Result[U, E]]) -> Result[U, E]:
    """flat_map: якщо Ok — викликати fn(value), якщо Err — коротке замикання."""
    if isinstance(r, Ok):
        return fn(r.value)
    return r  # type: ignore[return-value]


def map_err(r: Result[T, E], fn: Callable[[E], F]) -> Result[T, F]:
    """Перетворити тип помилки, не чіпаючи успіх."""
    if isinstance(r, Err):
        return Err(fn(r.error))
    return r  # type: ignore[return-value]


def unwrap_or(r: Result[T, E], default: T) -> T:
    """Дістати значення або повернути дефолт."""
    return r.value if isinstance(r, Ok) else default
```

3. Безпечний парсинг: parse_int, parse_pair.

lab09_result_maybe.py

```python
# C) Обгортання викликів, що кидають
# ===================================

def try_call(fn: Callable[..., T], *args: Any, **kwargs: Any) -> Result[T, str]:
    """Виклик функції зі спійманим винятком → Result[T, str]."""
    try:
        return Ok(fn(*args, **kwargs))
    except Exception as e:  # вузькі винятки — ще краще, але для демо так
        return Err(str(e))
```

4. **Доменний конвеєр з коротким замиканням: parse_pair → validate_age → calc_score → to_csv_row.**

lab09_result_maybe.py

```python
def pipeline_line(line: str) -> Result[str, str]:
    """line -> parse_pair -> validate_age -> calc_score -> to_csv_row, з коротким замиканням."""
    r: Result[tuple[str, int], str] = parse_pair(line)
    r = and_then(r, validate_age)
    r2: Result[User, str] = and_then(r, calc_score)
    r3: Result[str, str] = and_then(r2, to_csv_row)
    return r3
```

5. **Обробка списків: collect_results (зупинка на першій помилці), а також sequence/traverse.**

```python
# E) Обробка колекцій із зупинкою на першій помилці
# ==========================

def collect_results(items: Iterable[T], fn: Callable[[T], Result[U, E]]) -> Result[List[U], E]:
    acc: List[U] = []
    for it in items:
        r = fn(it)
        if isinstance(r, Err):
            return r
        acc.append(r.value)  # type: ignore[union-attr]
    return Ok(acc)


# Додатково: sequence/traverse (опційно, але корисно)
def sequence(rs: Iterable[Result[T, E]]) -> Result[List[T], E]:
    acc: List[T] = []
    for r in rs:
        if isinstance(r, Err):
            return r
        acc.append(r.value)  # type: ignore[union-attr]
    return Ok(acc)


def traverse(xs: Iterable[T], fn: Callable[[T], Result[U, E]]) -> Result[List[U], E]:
    return sequence(fn(x) for x in xs)
```

lab09_result_maybe.py

Результати:

```
(.venv) stanislav@fedora:~/Desktop/Functional_Programing/lab09$ python lab09_result_maybe.py
All tests are passed.
Demonstration: Err(error='Age out of range: 17')
```

Висновок: завдяки цій роботі я дізнався про те, що можна створювати програми-конвеєри, які не ламаються, а просто зупиняються. Це дозволяє легко з'єднувати багато кроків обробки даних, знаючи, що якщо один крок дасть помилку, всі наступні кроки автоматично пропустяться.